NEURAL NETWORKS FOR SIGNAL PROCESSING VII

PROCEEDINGS
OF THE 1997 IEEE
SIGNAL PROCESSING
SOCIETY WORKSHOP

Seventh in a Series of Workshops
Organized by the IEEE Signal Processing Society
Neural Networks Technical Committee

Edited by

Jose Principe
Lee Gile
Nelson Morgan
Elizabeth Wilson

Published under the sponsorship of the IEEE Signal Processing Society
in cooporation with the IEEE Neural Networks Council
and co-sponsored by the Air Force Office of Scientific Research (AFOSR)

The Institute of Electrical and Electronic Engineers, Inc.
New York, New York

1997 IEEE Workshop on Neural Networks for Signal Processing Proceedings

Copyright and Reprint Permission

IEEE Catalog Number	97TH8330
ISBN	0-7803-4256-9 (softbound) 0-7803-4257-7 (microfiche)
ISSN	1089-3555

Additional copies of this publication are available from

IEEE Operations Center
P.O. Box 1331
445 Hoes Lane
Piscataway, NJ 08855-1331 USA

1-800-678-IEEE
1-908-981-1393
1-908-981-9667 (FAX)
833-233 (Telex)
email: customer.services@ieee.org

The chapters in this book are based on presentations given at the IEEE Signal Processing Society Workshop on Neural Networks for Signal Processing.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave Blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	16 March 98	FINAL

4. TITLE AND SUBTITLE
1997 IEEE Workshop on Neural Networks for Signal Processing

5. FUNDING NUMBERS
G

6. AUTHORS

See Attached Proceedings

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Institute of Electrical and Electronics Engineers, Inc.
445 Hoes Lane, P.O. Box 1331
Piscataway, NJ 08855-1331

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Office of Scientific Research
110 Duncan Avenue, Room B115
Bolling AFB, DC 20332-8080

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
F49620-97-1-0311

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

NNSP'97 was held in Amelia Island Plantation, Amelia Island, Florida, September 24-26, 1997. NNSP'97 was sponsored by the Neural Networks Technical Committee of the IEEE Signal Processing Society, in cooperation with the IEEE Neural Network Council and with co-sponsorship from the Air Force Office of Scientific Research.

This year two topics were announced in the calls for papers. The goal was to create a critical mass of submissions and dedicate a full session to discuss a topic of current interest. This year topics were blind signal processing and pattern recognition applications. We had an exciting technical program with first rated papers and speakers.

We invited as plenary speakers Drs. Simon Haykin, from McMaster University, David Brown from F.D.A., S.Y. Kung, from Princeton, J. F. Cardoso, from Paris V, and Yann LeCun from ATT/BellLabs. Eighty high quality papers were presented with contributions from 16 countries. The proceedings were distributed at the conference, and were published by the IEEE Press. The organization sponsored ten students to present their work.

The highlights of the conference were the following: the advances made on blind source separation, both at the theoretical level and in practical applications; the advances in the implementation of the support vector machines; the interest in recurrent neural networks; the exciting new applications of neural networks to multimedia, and the continuing interest in biomedical applications. All the sessions were very lively, in particular the excellent poster sessions where lengthier discussions among researchers were conducted.

Overall this was a high quality conference, which everybody enjoyed greatly.

14. SUBJECT TERMS
Neural Networks
Pattern Recognition
Blind Source Separation

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	**Unclassified**	**Unclassified**	**UL**

UNIVERSITY OF
FLORIDA

Department of Electrical and Computer Engineering

458 ENG Bldg. 33
PO Box 116130
Gainesville, FL 32611-6130
(352) 392-2723 or (352) 392-2585
Fax (352) 392-0044

March 16, 1998

DTIC-OCP
8725 John J. Kingman Road Suite 0944
Fort Belvoir VA 20332-6218

RE: Grant Number: F49620-97-1-0311

SF298 Report Documentation Page-1997 IEEE Workshop on Neural Networks for Signal Processing

Two Proceedings of the 1997 IEEE Workshop Neural Networks for Signal Processing VII

I am enclosing the SF-298 Report Documentation Page for the final report and two proceedings on the above workshop that we held in Amelia Island Plantation, Amelia Island, Florida, September 24-26, 1997.

If there are any questions, please let me know.

Sincerely,

José C. Principe
BellSouth Professor
University of Florida
Department of Electrical and Computer Engineering
451 EB #33
POB 116130
Gainesville, FL 32611-6130
Phone: 352/392-2662
Fax: 352/392-0044
E-mail: principe@cnel.ufl.edu

Preface

This book contains referred papers presented at the Seventh IEEE Workshop on Neural Networks for Signal Processing (NNSP '97) held at the Amelia Island Plantation, Amelia Island, Florida on September 24–26, 1997.

The Neural Networks Technical Committee of the IEEE Signal Processing Society sponsored NNSP '97, in cooperation with the IEEE Neural Network Council and with co-sponsorship from the Air Force Office of Scientific Research (AFOSR). We designed the workshop to serve as a regular forum for researchers in academia and industry who are interested in the exciting field of neural networks for signal processing. Neural networks offer a fresh view for the important problems faced in signal processing because they extend linear models and go beyond the assumptions of stationarity and Gaussianity traditionally imposed in signal processing.

This year we announced two topics in the call for papers. The goal was to create a critical mass of submissions and dedicate a full session to discuss a topic of current interest. This year's topics are blind signal processing and biomedical applications. Each is important in its own right. Blind signal processing is a difficult but exciting area of signal processing with many practical applications for which the use of nonlinearity is key for acceptable solutions. The biomedical area has long been a challenging area due to the imprecise nature of human reasoning and the need for more sophisticated quantitative tools. Neural networks and other approximate reasoning methods are key players in this effort. We hope that this approach of electing topics will be successful and will make these proceedings a necessary reference for the advances reported in each field.

Our deep appreciation is extended to Drs. Simon Haykin, S.Y. Kung, J.F. Cardoso, Yann LeCun and David Brown for their insightful plenary talks. Our sincere thanks go to all members of the Technical Committee for the excellent and timely reviews, and above all to the authors whose contributions made this workshop possible.

Continuing with the tradition of paperless communication, this year's reviews and announcements were all electronic. Thanks to Dong-Wei Chen and Craig Fancourt for keeping the NNSP '97 Web page (http://www.cnel.ufl.edu/nnsp97/) current and effective. Special thanks go to Ms. Sharon Bosarge for her dedication and hard work to coordinate the many details necessary to put together the program and the local arrangements.

Jose C. Principe
Lee C. Giles
Bert DeVries
Nelson H. Morgan

NNSP '97 Technical Committee

Tulay Adali	University of Maryland Baltimore County
Charles M. Bachmann	Naval Research Laboratory
Andrew Back	RIKEN Laboratory
Sergio Carrato	University of Trieste
A. G. Constantinides	Imperial College
Federico Girosi	Massachusetts Institute of Technology
Allen L. Gorin	AT&T Laboratories
Lars Kai Hansen	Technical University of Denmark
John G. Harris	University of Florida
Yu-Hen Hu	University of Wisconsin - Madison
Jenq-Neng Hwang	University of Washington
Marc M. Van Hulle	University of Leuven
B. H. Juang	Lucent Technologies
Shigeru Katagiri	ATR Laboratories
Gary Kuhn	Siemens Corporate Research
S.Y. Kung	Princeton University
Jan Larsen	Technical University of Denmark
Richard Lippmann	MIT Lincoln Laboratory
Elias Manolakos	Northeastern University
John A. Sorensen	Technical University of Denmark
Volker Tresp	Siemens
Raymond Watrous	Siemens Corporate Research
Christian J. Wellekens	EURECOM

Table of Contents

THEORY

BIOMEDICAL

ALGORITHMS

BLIND SIGNAL SEPARATION

APPLICATIONS

THEORY

Invited Lecture

Chaos, radar clutter, and neural networks

Simon Haykin

McMaster University
Ontario, Canada

The lecture will be in three parts dealing in a coordinated way with the issues of chaos and neural networks. In the first part of the lecture, I will outline the set of criteria for determining if a given experimntal (physical) time series is indeed chaotic. The criteria include: Tests for nonlinearity, reliable estimations of the correlation dimension, characteristic time delay, embedding dimension, and the Liapunov spectrum.

In the second part of the lecture, I will present a case study based on real life data of sea clutter (i.e., radar backscatter from an ocean surface) and demonstrate how the above-mentioned criteria are satisfied in a very convincing way. The third part of the lecture addresses how a neural network can be used to peform dynamic reconstruction on an experimental time series known to be chaotic. The problem is usually complicated by the unavoidable presence of additive noise. For this part of the study, I will present the results of a detailed study involving the following learning algorithms:

Regularized radial basis function network.
Support vector machine.

The results of these evaluations will be checked against the chaos theory described in the first part of the lecture.

NONPARAMETRIC REGRESSION MODELING WITH TOPOGRAPHIC MAPS AS A BASIS FOR LOSSY IMAGE COMPRESSION

Marc M. Van Hulle
Laboratorium voor Neuro- en Psychofysiologie
K.U.Leuven
Campus Gasthuisberg
Herestraat
B-3000 Leuven, BELGIUM
Tel.: + 32 16 34 59 61 Fax: + 32 16 34 59 93
E-mail: marc@neuro.kuleuven.ac.be

Abstract— We introduce a new approach to lossy image compression with topographic maps based on nonparametric regression modeling: the topographic maps are trained to perform nonparametric regression using our recently introduced learning rule, called the Maximum Entropy learning Rule [9, 10], in combination with projection pursuit regression learning [11]. Furthermore, in order to better account for the local image statistics, we apply a technique similar to subspace classification. Finally, we compare the performance of our approach to that of the Karhunen-Loève transform and the optimally integrated adaptive learning algorithm [7].

INTRODUCTION

Kohonen's Self-Organizing (feature) Map (SOM) algorithm [1] is aimed at establishing, in an unsupervised way, a mapping from a higher dimensional space of input signals onto an equal or lower dimensional discrete lattice of formal neurons. The algorithm was originally conceived for nonparametric regression analysis, whereby the converged topographic map was intended to capture the main dimensions of the input space ([2], p. 152). Hence, the algorithm can be used for the regression modeling of multi-variable functions, at least in principle since it often yields "nonfunctional" mappings (one input can map onto more than one output, see [3, 4]). In addition to nonparametric regression, there also exists an intimate connection with the classic k-means clustering algorithm [12, 13] and the LBG algorithm [14] for building vector quantizers (except for the neighborhood function, see [15]). Hence, depending upon the interpretation adopted, the SOM algorithm performs regression modeling or vector quantization (VQ). In the VQ case, the SOM algorithm can be used as a "lossy" compression technique: the converged neuron weights form an optimal set of codewords –optimal with respect to the mean squared error (MSE) distortion due to quantization– and the min-

0-7803-4256-9/97/$10.00

imum Euclidean distance between an input sample and the neuron weights defines the code membership function. Such a VQ-based coding is radically different from a (linear) transform coding in which each input sample is projected along a limited number of projection directions –limited compared to the dimensionality of the input space. The optimal linear transformation is the Karhunen-Loève transform (KLT) since it minimizes the MSE [5] or, equivalently for this technique, since it maximizes the norm of the projected input vector.

The assumptions upon which the optimality conditions are based can be questioned, specifically the use of global statistics for generating an optimal coding scheme may not be appropriate when the input distribution is not stationary. In an attempt to remedy this problem, and to account for the local statistics as well, several adaptive techniques have been devised, also in the neural network field. The SOM algorithm was used as a basis of subspace classification [6]; the resulting structure is referred to as the adaptive subspace self-organizing algorithm [2]. More recently, Dony and Haykin [7] proposed an adaptive scheme, which favorably combined VQ with KLT, called the mixture of principal components (MPC). Specifically, it partitions the input distribution into a number of regions or classes as in VQ. Within each class, an input vector is represented by a linear combination of a limited number of basis vectors which define a subspace in a manner analogous to a principal components representation. The principal components are obtained with a neural network learning rule, such as Sanger's [8] or Oja's [6], since such an iterative approach to KLT requires less storage overhead and can be computationally more efficient than the algebraic approaches which operate on the sample covariance matrix directly [7]. The learning scheme as a whole is called the optimally integrated adaptive learning (OIAL) algorithm.

In this paper, we will introduce a novel way to perform image compression with neural networks: we will use topographic maps, trained for regression purposes, which become part of a technique similar to subspace classification. The topographic maps are trained with our recently introduced rule, called the Maximum Entropy learning Rule (MER) [9, 10], in combination with projection pursuit regression learning [11].

COMBINING PROJECTION PURSUIT REGRESSION WITH MER

Consider the regression fitting of a scalar function y of $d-1$ independent variables, denoted by the vector $\mathbf{x} = [x_1, ..., x_{d-1}]$, from a given set of M possibly noisy data points or measurements $\{(\mathbf{x}^m, y^m), m = 1, ..., M\}$ in d-dimensional space:

$$y^m = f(\mathbf{x}^m) + noise, \tag{1}$$

with f the unknown function to be estimated, and where the noise contribution has zero mean and is independent from the $\{\mathbf{x}^m\}$. In projection pursuit regression (PPR) [11], the d-dimensional data points are interpreted through optimally-chosen lower dimensional projections; the "pursuit" part refers to

optimization with respect to these projection directions. We will limit ourselves to the case where the function f is approximated by a sum of scalar functions f_k:

$$\hat{f}(\mathbf{x}) = \sum_{k=1}^{K} f_k(\mathbf{a}_k \mathbf{x}^T), \tag{2}$$

with $\hat{f}$ the approximation of f, and $\mathbf{a}_k$ the projection directions (unit vectors) and where T stands for transpose. The f_k are piecewise smooth activation functions or splines that join continuously at points called "knots." The functions f_k and projections $\mathbf{a}_k$ are estimated sequentially in order to minimize the mean squared error (MSE) of the residuals:

$$C(\mathbf{a}_k) = \frac{1}{M} \sum_{m=1}^{M} \left[f_k(\mathbf{a}_k \mathbf{x}^{m\ T}) - \left\{ y^m - \sum_{k'=1}^{k-1} f_{k'}(\mathbf{a}_{k'} \mathbf{x}^{m\ T}) \right\} \right]^2, \tag{3}$$

and where the term between the curly brackets denotes the kth residual of the mth data point, r_k^m, with $r_1^m \doteq y^m$. In other words, each newly-added projection $\mathbf{a}_k$ and activation function f_k are developed in the space of the residuals r_k^m which remain unaccounted for by the sum of the $k-1$ previously-determined activation functions.

If we consider the knots as neurons of a one-dimensional topographic map, then we can determine the position of these knots adaptively by developing the map in the two-dimensional space V of the inputs $\{(\mathbf{a}_k \mathbf{x}^{mT}), r_k^m\}$. As an example, consider the one-dimensional lattice shown in Fig. 1A. The neuron weights $\mathbf{w}_i = [w_{i1}, w_{i2}]$ will be updated with our previously developed learning rule, called the Maximum Entropy learning Rule (MER) [9, 10]. Before we introduce our rule, we first consider our definition of quantization region which is quite different from the one used in a Voronoi or Dirichlet tessellation of the input space, as done in the SOM algorithm. By observing that r_k is the dependent variable, each quantization region can be defined with respect to the projection direction $\mathbf{a}_k$ as the area demarcated by the $(\mathbf{a}_k \mathbf{x}^T)$-axis (horizontal) coordinates of two successive neurons of the lattice (thin dashed, vertical lines in Fig. 1A): *e.g.* quantization region H_c is bounded by the horizontal weight coordinates of neurons i and j. We associate with each quantization region a code membership function:

$$\mathbb{1}_{H_j}(\mathbf{a}_k \mathbf{x}^T) = \begin{cases} \frac{2^d}{n_{H_j}} & \text{if } \mathbf{a}_k \mathbf{x}^T \in H_j \\ 0 & \text{if } \mathbf{a}_k \mathbf{x}^T \notin H_j, \end{cases} \tag{4}$$

with n_{H_j}, $n_{H_j} \in \{1, 2\}$, the number of neurons that bound H_j, and $(\mathbf{a}_k \mathbf{x}^T)$ the projected input. (Note that since the quantization regions are bounded by the horizontal weight coordinates, their code membership functions only depend on the projected inputs). The Maximum Entropy learning Rule (MER) is defined as:

$$\Delta \mathbf{w}_i = \eta \sum_{H_j \in S_i} \mathbb{1}_{H_j}(\mathbf{a}_k \mathbf{x}^T)\ Sgn(\mathbf{v} - \mathbf{w}_i), \quad \forall i \in A, \tag{5}$$

with η the learning rate (a positive constant), $\mathbf{w}_i = [w_{i1}, w_{i2}]$ the weight vector of neuron i, $\mathbf{v} \doteq [(\mathbf{a}_k \mathbf{x}^T), r_k] \in V$ the current input vector, S_i the two quantization regions that have neuron i as a common bound, and $Sgn(.)$ the sign function taken componentwise. The effect of a single MER update is shown in Fig. 1A (thick dashed line). It can be formally proven that for a one-dimensional lattice, developed in one-dimensional space, MER yields an equiprobable quantization at convergence for any N [9], and that in the multi-dimensional case, MER yields a quantization which will approximate an equiprobable one at convergence for large N [10]. This implies that, *e.g.* for neuron j in Fig. 1B, there will be an equal number of data points below and above j (*i.e.* in the light and dark shaded regions). Since this is the case for every neuron of the lattice, there will be, roughly speaking, an equal number of data points from the set $\{(\mathbf{a}_k \mathbf{x}^{mT}, r_k^m)\}$ above and below the piecewise smooth activation function f_k (thick full line) at convergence. Furthermore, the neuron weights will represent the medians of the corresponding quantization regions: each weight vector will converge to the median, with the "median" defined as the vector of the (scalar) medians in each input dimension separately. (Note that there exists no unique definition of median for the higher-than-one dimensional case.) Finally, since a one-dimensional lattice is guaranteed to converge to an unfolded one in one-dimensional space [10], the lattice of Fig. 1A is guaranteed to converge to one which will always be unfolded with respect to the horizontal axis and, hence, we are assured to obtain a functional mapping.

Finally, the efficiency with which the input statistics is modeled can be improved by considering several regression models in parallel and treat them as classes. Each model can be trained with the aforementioned MER/PPR combination on common or on separate data sets. In case of the former, a class membership definition is needed. After training, data compression can be performed using the weights and projection directions of the trained regression models; model selection occurs according to the class membership function definition adopted for this application.

IMAGE COMPRESSION

Training of regression models

Consider a grey scale image $I(i, j)$ sized $M \times M$ pixels in which we select an $m \times m$ region or block (Fig. 2). The selected region is divided in two parts: 1) the central pixel at row i and column j, and 2) the surrounding pixels (shaded area), termed $surround(i, j)$. Consider now the regression fitting of the grey level of the central pixel as a function of the vector of grey levels of the surround: the pixels in the surround define an $m \times m - 1$ dimensional vector of independent variables and the central pixel represents the possibly noisy measurement of the unknown function which is to be estimated.

In order to better capture the local image statistics, we consider a total of L classes in parallel. Each class is represented by a regression model eq. (2) but with different projection directions and weight vectors. We divide the original image into L subimages sized $M \times M$ pixels. We assume a toroidal

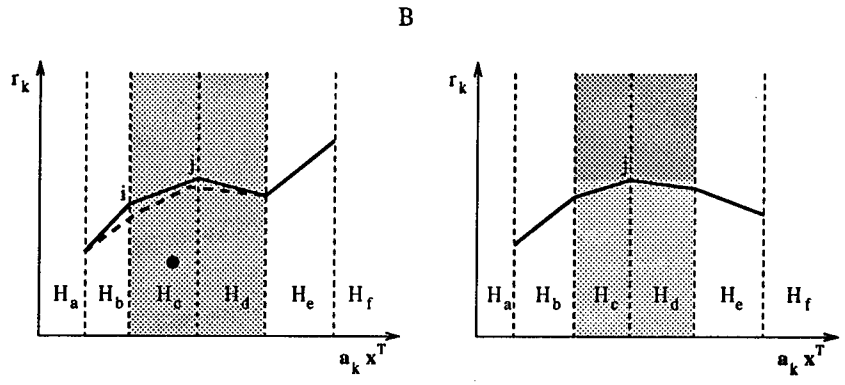


Figure 1: (A) Definition of quantization region in a one-dimensional lattice. The weights of the lattice are joined by line segments so as to yield the activation function f_k. The thick full and thick dashed lines represent f_k before and after the weights are updated once using MER, respectively. The thin dashed, vertical lines represent the borders of the quantization regions prior to updating the weights. The shaded region corresponds to the receptive field of neuron j and it comprises the quantization regions H_c and H_d. The present input is indicated by the black dot in H_c. (B) At convergence, there will be an equal number of data points in the dark and light shaded regions, below and above neuron j, respectively. The thick full line represents the piecewise linear activation function f_k.

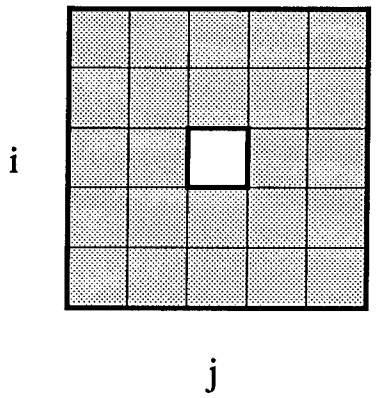


Figure 2: Definition of central pixel at coordinate (i, j) and the vector of surrounding pixels (shaded area), termed $surround(i, j)$.

extension for each subimage and a lateral shift of one pixel between two successive image blocks so that, in this way, we dispose of M^2 measurements per subimage in an $m \times m$ dimensional data space. Each regression model is trained on a different subimage until convergence.[1] The latter greatly simplifies the computational complexity of the algorithm, and the overall training time, and it alleviates the problem of how to properly initialize the regression models (see *e.g.* [7]).

Definition of codebook vector and class membership

Before we can perform image compression, we have to decide on the definitions of codebook vector and class membership. The latter will be similar but not equal to the one used in subspace classifiers. Assume that we use L classes of which the corresponding regression models consist of N neurons and K projection directions, for simplicity's sake. Each regression model is trained on blocks sized $m \times m$ pixels as explained in the previous paragraph. We now partition the image into nonoverlapping but juxtaposed blocks sized $m \times m$ pixels and project the $m \times m - 1$ dimensional surround vector of each block (*cf.* Fig. 2) along the K projection directions of each class. The nearest neuron weight vectors (in Euclidean sense), along each of the K projection directions, are then used for discretizing the projected surround vector and the binary indices of these weight vectors for storing the discretized surround vector; the central pixel value need not be discretized (and stored) since it can be predicted from the regression model of the selected class. The discretized surround vector can be computed from its K discretized projection coordinates. Hence, for each class, we store the $N \times K$ weight vectors from which the N^K codebook vectors needed for decompression can be determined.

Finally, for cases in which $L > 1$, we use the following definition of class membership: for each image block, we use the class which produces the smallest sum of the following two quantities: 1) the quantization error between the projected and the original surround vector, and 2) the regression error between the predicted and the original central pixel value. The binary code of the selected class is then stored together with the binary code of the K discretized projection coordinates. Decompression is achieved when the binary codes are substituted for the corresponding codebook vectors; the central pixel value is obtained from the corresponding regression model. The image obtained in this way is termed the decompressed image. The quality of the decompressed image can be improved when, for $L = 1$, the regression model is applied to the decompressed image but now with overlapping image blocks. For $L > 1$, we can take a conservative approach by selecting the regression model which yields the smallest discrepancy between the central pixel value it produces and the one we had before in the decompressed image.

[1] Since we do not treat the subimages as preclassified data, we also considered the alternative case where a single training set is used and where training of each one of the L regression models continues on those training samples for which it yields the smallest regression errors until all samples are consistently represented. However, since this recursive approach yielded only a minor improvement for the case reported in the Simulation Results section, but at the expense of a much increased computational complexity and training time, we did not consider it further.

SIMULATION RESULTS

In the simulations, we consider 8 bit images and surrounds sized $5\times5-1$ pixels (*i.e.* $m = 5$). We use lattices of N neurons, with cubic spline interpolation between the neuron weights, K projection directions and L regression models in such a way that $(N+1) \times K \times L = 64$ (we use $(N+1)$ to account for the space needed for storing the codebook vectors). The compression ratio can be estimated as follows: for the original image we need 8 bits per pixel (*bpp*) and for the compressed image we need $(N+1) \times K \times L$ values or 6 bits to code for the vector of grey values of each $m \times m$ block in which the image is partitioned, *i.e.* 0.24 *bpp*.

We run MER in batch mode ($\eta = 0.02$) and determine, after each epoch, the MSE between the actual and the desired, equiprobable code membership function usage. We run MER until the magnitude of the difference between the present and the previous running-averaged MSE is lower than $1.0\ 10^{-7}$ or until 15,000 epochs have elapsed; the present running average equals 10% of the present, unaveraged MSE added to 90% of the previous running average. In order to optimize $C(\mathbf{a}_k)$, the procedure is run for the $\mathbf{a}_k$ taken as unit vectors; the components of the unit vector with the lowest residual error are then further optimized by performing hill descent on $C(\mathbf{a}_k)$ in steps of 0.01; after each update, $\mathbf{a}_k$ is renormalized to unit length. Finally, in order to further optimize the K projection directions obtained, we apply backfitting: we cyclically minimize $C(\mathbf{a}_K)$ for the residuals of projection direction k, until there is little or no change ($< 0.1\%$) or until 10 full cycles have elapsed. We also ensure that, after each update, $\mathbf{a}_k$ is renormalized.

As an example, we consider the LENA image (Fig. 3A), sized 512×512 pixels at 8 bits per pixel. We modify the grey scale of the image from $[0, 255]$ to $[0, 10]$ and select L subimages sized 128×128 pixels (Fig. 3B) and use the 16,384 data vectors corresponding to each subimage for training the lattices as explained in the previous section. For the time being, we have considered only $L = 1$, 2 and 4. When $L = 1$ only subimage 1 is used, when $L = 2$ subimages 1 and 2 are used, and so on. In order to quantify the simulation results, we compute the MSE (MSE_c) and the signal-to-noise-ratio due to quantization between the original and the decompressed image:

$$SNR_c = 10 \log_{10} \frac{\sum_{i,j} I(i,j)^2}{\sum_{i,j} (\hat{I}(i,j) - I(i,j))^2}, \quad \text{(dB)} \tag{6}$$

with $\hat{I}$ the decompressed image. The results for the LENA image are summarized in Table 1; the second column lists the actual configuration used, abbreviated symbolically as $L/N/K$; the last two columns list the corresponding MSE_c and SNR_c values. We observe that the performance of MER/PPR improves when N is lowered from 15 to 7 so that L and K can increase. The decompressed image for $L/N/K = 2/7/4$ is shown in Fig. 3C. Using the same standard image, the results obtained with the OIAL and KLT algorithms are also shown in Table 1 [16]. The OIAL algorithm uses $K = 4$ projections and $L = 128$ classes and the KLT algorithm uses $K = 4$ projections only, both for 0.25 *bpp* but for blocks sized 8×8 pixels. We observe that MER/PPR performs reasonably well for $N = 7$, however, we hasten to

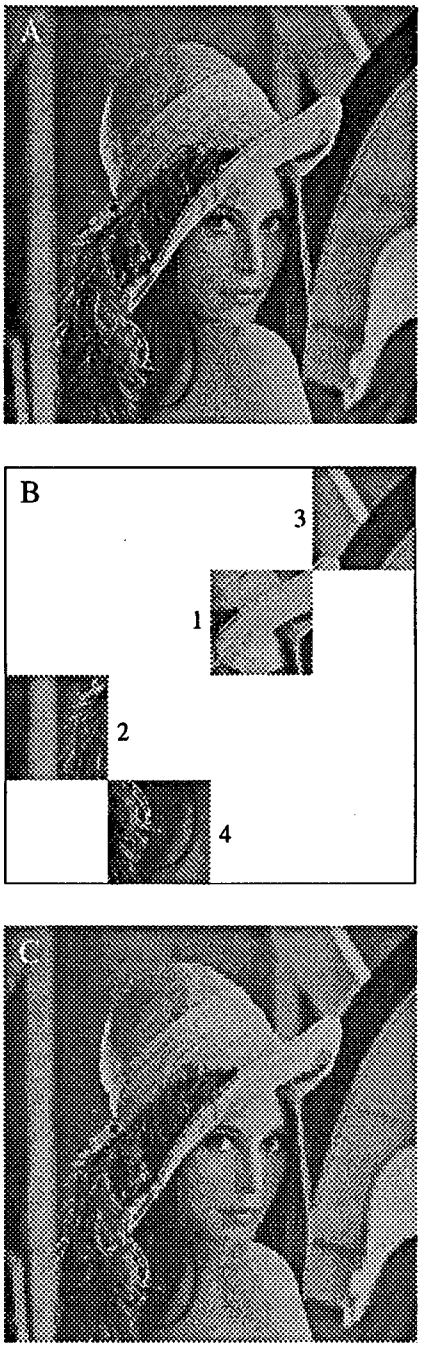


Figure 3: (A) Original LENA image, (B) subimages used for training, and (C) decompressed image obtained with MER/PPR ($L/N/K = 2/7/4$).

Table 1: Performances obtained with the MER/PPR, OIAL and KLT algorithms

algorithm	$L/N/K$	MSE_c	SNR_c
MER/PPR	1/7/8	82.6	23.0
	2/7/4	56.7	24.6
	4/7/2	69.8	23.8
	1/15/4	89.7	22.7
	2/15/2	91.1	22.6
OIAL	128/-/4	54.9	24.8
KLT	1/-/4	71.0	23.7

add that much more images should be considered before any judgment can be made.

CONCLUSION

In this paper, we have introduced a new approach to lossy image compression using topographic maps and a technique similar to subspace classification. Within each class, a number of topographic maps were trained so that a nonparametric regression model of the local image statistics is obtained. The topographic maps were trained using our recently introduced rule, called the Maximum Entropy learning Rule (MER) [9, 10], in combination with projection pursuit regression (PPR) learning [11]. The use of PPR in combination with MER offers the advantage that we don't need a prohibitive amount of neurons for regression modeling in high dimensional spaces and a high number of input samples to allocate the neuron weights reliably (*cf.* the curse of dimensionality). Furthermore, and in particular with respect to small data sets, since with MER the neuron weights converge to the medians of their quantization regions, the regression models obtained will be less sensitive to outliers but, on the other hand, they will be more sensitive to biased noise. Fortunately, the effect of the latter can be reduced by backfitting. Finally, since we have essentially trained our topographic maps for nonparametric regression purposes, we could equally well have considered noise cancelling as an application. This will be explored in our future work.

ACKNOWLEDGEMENTS

The author is a research associate of the Fund for Scientific Research - Flanders (Belgium) and is supported by research grants received from the Fund for Scientific Research (G.0185.96), the Research Fund of the K.U.Leuven (F/95/138) and the European Commission (ECVnet EP8212).

REFERENCES

[1] T. Kohonen, "Self-organized formation of topologically correct feature maps," Biol. Cybern., vol. 43, 1982, pp. 59-69.

[2] T. Kohonen, "Self-organizing maps," Heidelberg: Springer, 1995.

[3] H. Ritter, and K. Schulten, "Combining self-organizing maps," Proc. Intl Joint Conference on Neural Networks 2, pp. 499-502, 1989.

[4] V. Cherkassky, and H. Lari-Najafi, "Constrained Topological Mapping for Nonparametric Regression Analysis," Neural Networks, vol. 4, 1991, pp. 27-40.

[5] R.C. Gonzales, and R.E. Woods, "Digital image processing," Reading Mass: Addison-Wesley, 1993.

[6] E. Oja, "Neural networks, principal components, and subspaces," International Journal of Neural Systems, vol. 1, 1989, pp. 61-68.

[7] R.D. Dony, and S. Haykin, "Optimally Adaptive Transform Coding," IEEE Trans. on Image Processing, vol. 4(10), 1995, pp. 1358-1370.

[8] T.D. Sanger, "Optimal unsupervised learning in a single-layer linear feedforward neural network," Neural Networks, vol. 2, 1989, pp. 459-473.

[9] M.M. Van Hulle, "Globally-ordered topology-preserving maps achieved with a learning rule performing local weight updates only," Proc. IEEE NNSP95 (Cambridge, MA 1995), pp. 95-104, 1995.

[10] M.M. Van Hulle, "The formation of topographic maps which maximize the average mutual information of the output responses to noiseless input signals," Neural Computation, vol. 9(3), 1997, pp. 595-606.

[11] J.H. Friedman, and W. Stuetzle, "Projection pursuit regression," Journal of the American Statistical Association, vol. 76(376), 1981, pp. 817-823.

[12] P.R Krishnaiah, and L.N. Kanal, eds., "Classification, Pattern Recognition, and Reduction of Dimensionality." Handbook of Statistics, vol. 2, Amsterdam: North Holland, 1982.

[13] J. Hertz, A. Krogh, and R.G. Palmer, "Introduction to the theory of neural computation," Reading MA: Addison-Wesley, 1991.

[14] Y. Linde, A. Buzo, and R.M. Gray, "An Algorithm for Vector Quantizer Design. IEEE Trans. on Communications, vol. COM-28, pp. 84-95, 1980.

[15] S.P. Luttrell, "Code vector density in topographic mappings: scalar case," IEEE Trans. on Neural Networks, vol. 2, pp. 427-436, 1991.

[16] S. Haykin, "Neural networks expand SP's horizons," IEEE Signal Processing Magazine, March 1996, pp. 24-49.

Entropy Manipulation of Arbitrary Nonlinear Mappings

John W. Fisher III **José C. Principe**

Computational NeuroEngineering Laboratory
EB, #33, PO Box 116130
University of Floridaa
Gainesville, FL 32611-6130
jwf@ecl.ufl.edu
principe@cnel.ufl.edu

Abstract

We discuss an unsupervised learning method which is driven by an information theoretic based criterion. Information theoretic based learning has been examined by several authors Linsker [2, 3], Bell and Sejnowski [5], Deco and Obradovic [1], and Viola *et al* [6]. The method we discuss differs from previous work in that it is extensible to a feed-forward multi-layer perceptron with an arbitrary number of layers and makes no assumption about the underlying PDF of the input space. We show a simple unsupervised method by which multi-dimensional signals can be nonlinearly transformed onto a maximum entropy feature space resulting in statistically independent features.

1.0 INTRODUCTION

Our goal is to develop mappings that yield statistically independent features. We present here a nonlinear adaptive method of feature extraction. It is based on concepts from information theory, namely mutual information and maximum cross-entropy. The adaptation is unsupervised in the sense that the mapping is determined without assigning an explicit target output, à priori, to each exemplar. It is driven, instead, by a global property of the output: cross entropy.

There are many mappings by which statistically independent outputs can be obtained. At issue is the usefulness of the derived features. Towards this goal we apply Linsker's *Principle of Information Maximization* which seeks to transfer maximum *information* about the input signal to the output features. It is also shown that the resulting adaptation rule fits naturally into the back-propagation method for training multi-layer perceptrons.

Previous methods [1] have optimized entropy at the output of the mapping by considering the underlying distribution at the input. This represents a complex problem for general nonlinear mappings. The method presented here, by contrast, is more directly related to the technique of Bell and Sejnowski [5] in which we manipulate

0-7803-4256-9/97/$10.00

entropy through observation at the output of the mapping. Specifically, we exploit a property of entropy coupled with a saturating nonlinearity which results in a method for entropy manipulation that is extensible to feed-forward multi-layer perceptrons (MLP). The technique can be used for an MLP with an arbitrary number of hidden layers. As mutual information is a function of two entropy terms, the method can be applied to the manipulation of mutual information as well.

In section 2 we discuss the concepts upon which our feature extraction method is based. We derive the adaptation method which results in statistically independent features in section 3. An example result is presented in section 4, while our conclusions and observations appear in section 5.

2.0 BACKGROUND

The method we describe here combines cross entropy maximization with Parzen window probability density function estimation. These concepts are reviewed.

2.1 Maximum Entropy as a Self-organizing Principle

Maximum entropy techniques have been applied to a host of problems (e.g. blind separation, parameter estimation, coding theory, etc.). Linsker [2] proposed maximum entropy as a self-organizing principle for neural systems. The basic premise being that any mapping of a signal through a neural network should be accomplished so as to maximize the amount of information preserved. Linsker demonstrates this *principle of maximum information preservation* for several problems including a deterministic signal corrupted by gaussian noise. Mathematically Linsker's principle is stated

$$I(x, y) = h_Y(y) - h_{Y|X}(y|x) \tag{1}$$

where $I(x, y)$ is the mutual information of the RVs X and Y, and $h([\,])$ is the continuous entropy measure [4]. Given the RV (random vector), $Y \in \Re^N$, the continuous entropy is defined as

$$h_Y(u) = -\int_{-\infty}^{\infty} \log(f_Y(u)) f_Y(u) du\,, \tag{2}$$

where $f_Y(u)$ is the probability density function of the RV, the base of the logarithm is arbitrary, and the integral is N-fold. Several properties of the continuous entropy measure are of interest.

1. If the RV is restricted to a finite range in $\Re^N$ the continuous entropy measure is maximized for the *uniform* distribution.
2. If the covariance matrix is held constant the measure is maximized for the *normal* distribution.

3. If the RV is transformed by a mapping $g:\Re^N \rightarrow \Re^N$ then the entropy of the new RV, $y = g(x)$, satisfies the inequality

$$h_Y(y) \leq h_X(x) + E\{\ln(|J_{XY}|)\}, \tag{3}$$

with equality if and only if the mapping has a unique inverse, where J_{XY} is the Jacobian of the mapping from X to Y.

Regarding the first two properties we note that for either case each element of the RV is *statistically independent* from the other elements.

Examination of (3) implies that by transforming a RV we can increase the amount of information. This is a consequence of working with continuous RVs. In general the continuous entropy measure is used to compare the relative entropies of several RVs. We can see from (3), that if two RVs are mapped by the same invertible *linear* transformation their relative entropies (as measured by the difference) remains unchanged. However, if the mapping is nonlinear, in which case the second term of (3), is a function of the random variable, it is possible to change relative information of two random variables. From the perspective of classification this is an important point. If the mapping is *topological* (in which case it has a unique inverse), there is no increase, theoretically, in the ability to separate classes. That is, we can always reflect a discriminant function in the transformed space as a warping of another discriminant function in the original space. However, finding the discriminant function is a different problem altogether. By changing the relative information, the form of the discriminant function may be simpler.

This is not true, however, for a mapping onto a subspace. Our implicit assumption here is that we are unable to reliably determine a discriminant function in the full input space. As a consequence we seek a subspace mapping that is in some measure optimal for classification. We cannot avoid the loss of information (and hence some ability to discriminate classes) when using a subspace mapping. However, if the criterion used for adapting the mapping, is entropy based, we can perhaps minimize this loss. It should be mentioned that in *all* classification problems there is an implicit assumption that the classes to be discriminated do indeed lie in a subspace.

2.2 Nonparametric Pdf Estimation

One difficulty in applying the continuous entropy measure with continuous RVs is that it requires some knowledge of the underlying PDF (probability distribution function). Unless assumptions are made about the form of the density function it is very difficult to use the measure directly. A nonparametric kernel-based method for estimating the PDF is the Parzen window method [7]. The Parzen window estimate of the probability distribution, $\hat{f}_Y(u)$, of a random vector $Y \in \Re^N$ at a point u is defined as

$$\hat{f}_Y(u) = \left(\frac{1}{N_y}\right) \sum_{i=1}^{N_y} \kappa(y_i - u). \tag{4}$$

The vectors $y_i \in \Re^N$ are observations of the random vector and $\kappa([\,])$ is a kernel function which itself satisfies the properties of PDFs (i.e. $\kappa(u) > 0$ and $\int \kappa(u)du = 1$). Since we wish to make a local estimate of the PDF, the kernel function should also be localized (i.e. uni-modal, decaying to zero). In the method we describe we will also require that $\kappa([\,])$ be differentiable everywhere. In the multidimensional case the form of the kernel is typically gaussian or uniform. As a result of the differentiability requirement, the gaussian kernel is most suitable here. The computational complexity of the estimator increases with dimension, however, we will be estimating the PDF in the output space of our multi-layer perceptron where the dimensionality can be controlled.

3.0 DERIVATION OF LEARNING ALGORITHM

As we stated our goal is to find statistically independent features; features that jointly posses minimum mutual information or maximum cross entropy.

Suppose we have a mapping $g:\Re^N \rightarrow \Re^M$; $M < N$, of a random vector $X \in \Re^N$, which is described by the following equation

$$Y = g(\alpha, X) \tag{5}$$

How do we adapt the parameters α such that the mapping results in a maximum cross-entropy random variable? If we have a desired target distribution then we can use the Parzen windows estimate to minimize the "distance" between the observed distribution and the desired distribution. If the mapping has a restricted range (as does the output of an MLP using sigmoidal nonlinearities), the uniform distribution (which has maximum entropy for restricted range) can be used as the target distribution. If we adapt the parameters, α, of our mapping such that the output distribution is uniform, then we will have achieved statistically independent features regardless of the underlying input distribution.

Viola *et al* [6] has taken a very similar approach to entropy manipulation, although that work differs in that it does not address nonlinear mappings directly, the gradient method is estimated stochastically, and entropy is worked with explicitly. By our choice of topology (MLP) and distance metric we are able to work with entropy indirectly and fit the approach naturally into a back-propagation learning paradigm.

As our minimization criterion we use integrated squared error between our estimate and the desired distribution, which we approximate with a summation.

$$\begin{aligned} J &= \frac{1}{2}\int_{\Omega_Y} \left(f_Y(u) - \hat{f}_Y(u,y)\right)^2 du \\ &\approx \sum_j \frac{1}{2}\left(f_Y(u_j) - \hat{f}_Y(u_j,y)\right)^2 \Delta u \qquad y = \{y_1 \ldots y_{N_Y}\} \end{aligned} \tag{6}$$

In (6), Ω_Y indicates the nonzero region (a hypercube for the uniform distribution) over which the M-fold integration is evaluated. The criterion above exploits the fact that the MLP with saturating nonlinearities has finite support at the output. This

fact coupled with property 1 (i.e. as the integrated squared error between the observed output distribution and the uniform distribution is minimized, entropy is maximized) makes the criterion suitable for entropy manipulation.

Assuming the output distribution is sampled adequately, we can approximate this integral with a summation in which $u_j \in \Re^M$ are samples in M-space and Δu is represents a volume.

The gradient of the criterion function with respect to the mapping parameters is determined via the chain rule as

$$\begin{aligned}\frac{\partial J}{\partial \alpha} &= \left(\frac{\partial J}{\partial \hat{f}}\right)\left(\frac{\partial \hat{f}}{\partial g}\right)\left(\frac{\partial g}{\partial \alpha}\right) \\ &= \left(\frac{\Delta u}{N_u}\right)\sum_j (f_Y(u_j) - \hat{f}_Y(u_j, y))\left(\frac{\partial \hat{f}}{\partial g}\right)\left(\frac{\partial g}{\partial \alpha}\right), \\ &= \left(\frac{\Delta u}{N_u}\right)\sum_j \varepsilon_Y(u_j, y)\left(\frac{\partial \hat{f}}{\partial g}\right)\left(\frac{\partial g}{\partial \alpha}\right)\end{aligned} \tag{7}$$

where $\varepsilon_Y(u_j, y)$ is the computed distribution error over all observations y. The last term in (7), $\partial g / \partial \alpha$, is recognized as the sensitivity of our mapping to the parameters α. Since our mapping is a feed-forward MLP (α represents the weights and bias terms of the neural network), this term can be computed using standard backpropagation. The remaining partial derivative, $\partial \hat{f} / \partial g$, is

$$\begin{aligned}\frac{\partial \hat{f}}{\partial g} &= \left(\frac{1}{N_Y}\right)\sum_{i=1}^{N_Y} \kappa'(y_i - u_j) \\ &= \left(\frac{1}{N_Y}\right)\sum_{i=1}^{N_Y} \kappa'(g(\alpha, x_i) - u_j)\end{aligned} \tag{8}$$

Substituting (8) into (7) yields

$$\frac{\partial J}{\partial \alpha} = \left(\frac{\Delta u}{N_u N_Y}\right)\sum_j \sum_i \varepsilon_Y(u_j, y_i)\kappa'(g(\alpha, x_i) - u_j)\left(\frac{\partial}{\partial \alpha} g(\alpha, x_i)\right) \tag{9}$$

The terms in (9), excluding the mapping sensitivities, become the new error term in our backpropagation algorithm. This adaptation scheme is depicted in figure 1, which shows that this adaptation scheme fits neatly into the backpropagation paradigm.

Examination of the gaussian kernel and its differential in two dimension illustrates some of the practical issues of implementing this method of feature extraction as well as providing an intuitive understanding of what is happening during the adap-

tation process. The N-dimensional gaussian kernel evaluated at some u is (simplified for two dimensions)

$$\begin{aligned}\kappa(y_i-u) &= \frac{1}{(2\pi)^{N/2}|\Sigma|^{1/2}}\exp\left(-\frac{1}{2}(y_i-u)^\dagger\Sigma^{-1}(y_i-u)\right) \\ &= \frac{1}{2\pi\sigma^2}\exp\left(-\frac{1}{2\sigma^2}y_i-u^\dagger y_i-u\right) \quad ; \ \Sigma = \begin{bmatrix}\sigma^2 & 0\\ 0 & \sigma^2\end{bmatrix}, N=2\end{aligned} \tag{10}$$

The partial derivative of the kernel (also simplified for the two-dimensional case) with respect to the input y_i as observed at the output of the MLP is

$$\begin{aligned}\frac{\partial\kappa}{\partial y_i} &= -\left(\frac{\exp\left(-\frac{1}{2}(y_i-u)^\dagger\Sigma^{-1}(y_i-u)\right)}{(2\pi)^{N/2}|\Sigma|^{1/2}}\right)\Sigma^{-1}(y_i-u) \\ &= \kappa(y_i-u)\Sigma^{-1}(u-y_i) \\ &= \left(\frac{\exp\left(-\frac{1}{2\sigma^2}(y_i-u)^\dagger(y_i-u)\right)}{2\pi\sigma^4}\right)(u-y_i) \quad ; \ \Sigma = \begin{bmatrix}\sigma^2 & 0\\ 0 & \sigma^2\end{bmatrix}, N=2\end{aligned} \tag{11}$$

These functions are shown in figure 2. The contour of the gaussian kernel is useful in that it shows that output samples, y_i, greater than two standard deviations from the center, u, of the kernel (in the feature space) do not significantly impact the estimate of the output PDF at that sample point. Likewise, the gradient term, is not significant for output samples exceeding two standard deviations from the kernel center. Consequently sample points for the PDF estimate should not exceed a distance of two standard deviations from each other, otherwise, samples caught "in between" do not contribute significantly to the estimate of the PDF. A large number of such samples can cause very slow adaptation.

Recall that the terms in (9) replace the standard error term in the backpropagation algorithm. This term is plotted as a surface in figure 2 minus the PDF error. From this plot we see that the kernels act as either local attractors or repellors depending on whether the computed PDF error is negative (repellor) or positive (attractor). In this way the adaptation procedure operates in the feature space locally from a globally derived measure of the output space (PDF estimate).

4.0 EXPERIMENTAL RESULTS

We have conducted experiments using this method on millimeter-wave ISAR (inverse synthetic aperture radar) images (64 x 64 pixels). The mapping structure we use in our experiment is a multi-layer perceptron with a single hidden layer (4096 input nodes,4 hidden nodes, 2 output nodes). Using the adaptation method

described, we trained the network on two vehicle types with ISAR images from 180 degrees of aspect. The projection of the training images (and between aspect testing images) is shown in figure 3 (where adjacent aspect training images are connected). As can be some significant class separation is exhibited (without prior labeling of the classes). We also note that the points where the classes overlap correspond to the cardinal aspect angles, which are, in general, difficult aspect angles to separate on similar vehicles in this type of imagery.

5.0 CONCLUSIONS

We have presented what we believe to be a new method of unsupervised learning. This method unlike previous methods is not limited to linear topologies [3] nor unimodal PDFs [5]. In effect, we achieve features which are statistically independent from each other and yet are still, clearly, structurally related to the input structure as exhibited by the results of our example. This property bears similarity to Kohonen's discrete SOFM, however our map exists in a continuous output space. We are pursuing in our research more rigorous analysis in the comparison of the resulting feature maps to the Kohonen type. We are utilizing this method as a preprocessing for classification in our continuing research, although other applications certainly exist (e.g. blind separation).

Acknowledgments

This work was partially supported by DARPA grant F33615-97-1-1019.

REFERENCES

[1] G. Deco and D. Obradovic, 1996, *An Information-Theoretic Approach to Neural Computing*, Springer-Verlag, New York

[2] R. Linsker, 1988, "Self-organization in a perceptual system.", *Computer,* vol. 21, pp. 105-117.

[3] R. Linsker, 1990, "How to generate ordered maps by maximizing the mutual information between input and output signals.", *Neural Computation,* 1, 402-411.

[4] A. Papoulis, 1991, *Probability, Random Variables, and Stochastic Processes*, 3rd Ed, pp. 533-602, McGraw-Hill.

[5] T. Bell and J. Sejnowski, 1995, "An Information-Maximization Approach to Blind Separation and Blind Deconvolution", *Neural Computation* 7, 1129-1159.

[6] P. Viola, N. Schraudolph, and J. Sejnowski, 1996, "Empirical Entropy Manipulation for Real-World Problems", *Neural Information Processing Systems* 8, MIT Press, 1996.

[7] E. Parzen, 1962, "On the estimation of a probability density function and the mode.", *Ann. Math. Stat.,* 33, pp. 1065-1076.

[8] T. Kohonen, 1988, *Self-Organization and Associative Memory* (1st ed.), Springer Series in Information Sciences, vol. 8, Springer-Verlag.

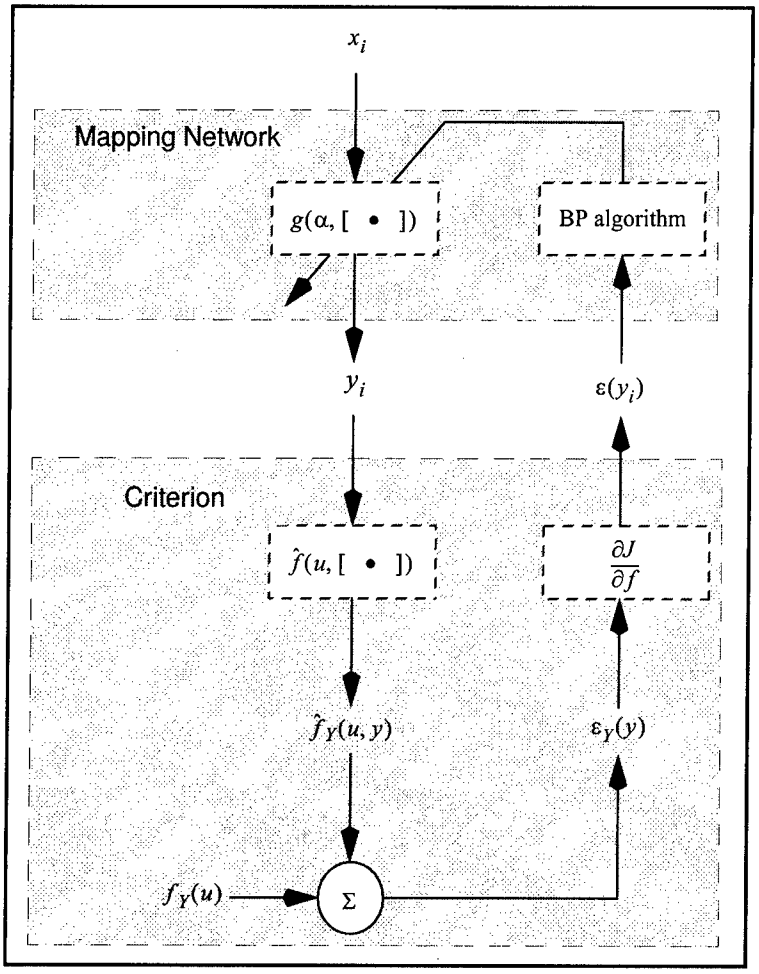


Figure 1 Block diagram of PDF driven adaptation scheme

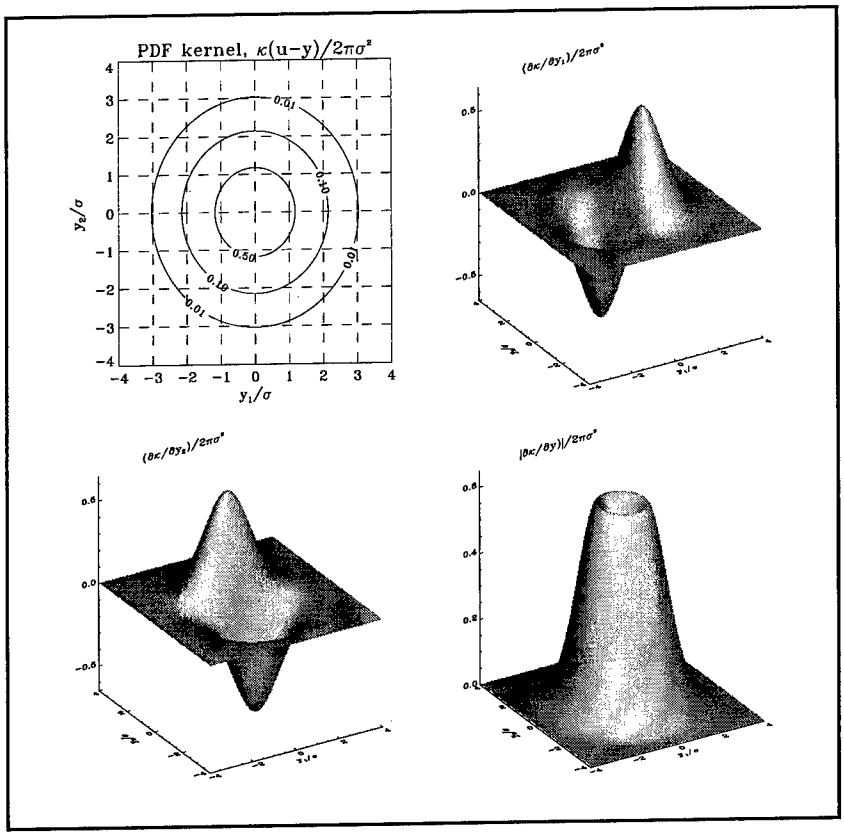


Figure 2 The plots above assume that we are using a two-dimensional gaussian kernel with a diagonal covariance matrix with σ^2 on the diagonals. Contour of the gaussian kernel (top left, normalized by σ), surface plots of the gradient terms with respect to y_1 (top right), y_2 (bottom left), and magnitude (bottom right) all normalized by σ^3. These terms are essentially zero at a distance of two standard deviations.

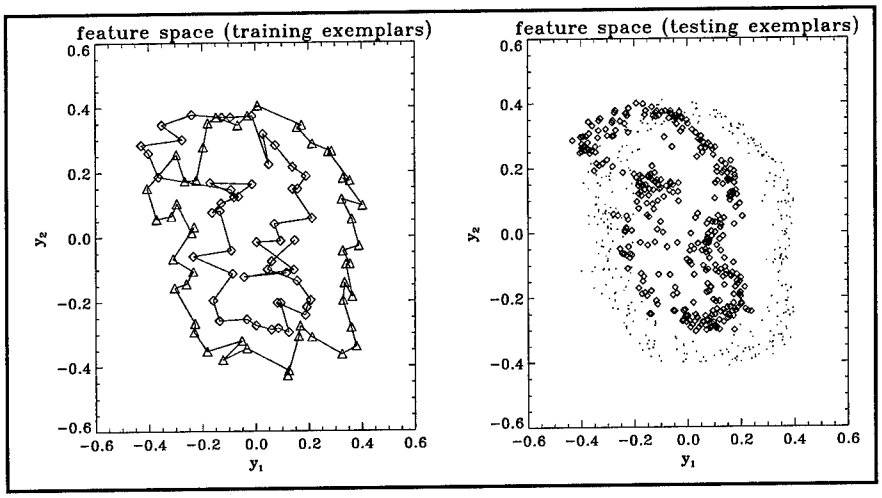


Figure 3 Example of training on ISAR images of two vehicles (aspect varying over 180 degrees). Over most of the aspect angles the vehicles are separated in the new feature space. Adjacent aspect angles are connected in the training set, evidence that topological neighborhoods were maintained. The mapping also generalizes to the testing set as well.

ADAPTIVE REGULARIZATION OF NEURAL CLASSIFIERS

L. Nonboe Andersen, J. Larsen, L.K. Hansen & M. Hintz-Madsen
CONNECT, Department of Mathematical Modelling, Building 321
Technical University of Denmark, DK-2800 Lyngby, Denmark
Phones: +45 4525+ext. 3899,3923,3889,3894
Fax: +45 45872599
emails: lna,jl,lkhansen,mhm@imm.dtu.dk
www: http://eivind.imm.dtu.dk

Abstract. In this paper we present a regularization scheme which iteratively adapts the regularization parameters by minimizing the validation error. It is suggested to use the adaptive regularization scheme in conjunction with Optimal Brain Damage pruning to optimize the architecture and to avoid overfitting. Furthermore, we propose an improved neural classification architecture eliminating an inherent redundancy in the widely used SoftMax classification network. Numerical results demonstrate the viability of the method.

INTRODUCTION

Neural networks are flexible tools for pattern recognition and by expanding the network architecture any relevant target function can be approximated [6]. In this contribution we present an improved version of the neural classifier architecture based on a feed-forward net with SoftMax [2] normalization presented in [7], [8] avoiding an inherent redundant parameterization. The outputs of the network estimate the class conditional posterior probabilities and the network is trained using a maximum a posteriori (MAP) framework.

The associated risk of overfitting on noisy data is of major concern in neural network design [4]. The objective of architecture optimization is to minimize the generalization error. The architecture can be optimized *directly* by e.g., pruning techniques or *indirectly* by using regularization. One might consider various regularization schemes: from adapting a single regularization parameter to individual regularization of the weights in the net. These subjects are further addressed in [9], [10]. We suggest a hybrid approach with Optimal Brain Damage [11] for pruning and an adaptive regularization scheme. The inevitable problem of adapting the *amount* of regularization is solved by minimizing the generalization error w.r.t. regularization parameters. Using the validation error calculated from a single validation set as an

0-7803-4256-9/97/$10.00

estimate of the generalization error, it is possible to formulate an iterative gradient descent scheme for adapting the regularization parameters [9]. The Bayesian way to adapt regularization parameters is to minimize the evidence [1, Ch. 10], [14]; however, the evidence does not, in a simple way, relate to the generalization error which is our primary object of interest.

NETWORK ARCHITECTURE

Suppose that the input (feature) vector is denoted by $\boldsymbol{x}$ with $\dim(\boldsymbol{x}) = n_I$. The aim is to model the posterior probabilities $p(\mathcal{C}_i|\boldsymbol{x})$, $i = 1, 2, \cdots, c$ where $\mathcal{C}_i$ denotes the i'th class. Then under a simple loss function the Bayes optimal[1] classifier assigns class label $\mathcal{C}_i$ to $\boldsymbol{x}$ if $i = \arg\max_j p(\mathcal{C}_j|\boldsymbol{x})$.

Following [8] (see also [1]), the outputs, $\widehat{y}_i$, of the neural network represent *estimates* of the posterior probabilities, i.e., $\widehat{y}_i = \widehat{p}(\mathcal{C}_i|\boldsymbol{x})$; hence, $\sum_{i=1}^{c} p(\mathcal{C}_i|\boldsymbol{x}) = 1$. That is, we need merely to estimate $c-1$ posterior probabilities, say $p(\mathcal{C}_i|\boldsymbol{x})$, $i = 1, 2, \cdots, c-1$, then the last is calculated as $p(\mathcal{C}_c|\boldsymbol{x}) = 1 - \sum_{i=1}^{c-1} p(\mathcal{C}_i|\boldsymbol{x})$.

Define a 2-layer feed-forward network with n_I inputs, n_H hidden neurons and $c-1$ outputs by:

$$h_j(\boldsymbol{x}) = \tanh\left(\sum_{\ell=1}^{n_I} w_{j\ell}^I x_\ell + w_{j0}^I\right), \quad \phi_i(\boldsymbol{x}) = \sum_{j=1}^{n_H} w_{ij}^H h_j(\boldsymbol{x}) + w_{i0}^H \qquad (1)$$

where $w_{j\ell}^I$, w_{ij}^H are the input-to-hidden and hidden-to-output weights, respectively. All weights are assembled in the weight vector $\boldsymbol{w} = \{w_{j\ell}^I, w_{ij}^H\}$.

In order to interpret the network outputs as probabilities a *modified* normalized exponential transformation similar to SoftMax [2] is used,

$$\widehat{y}_i = \frac{\exp(\phi_i)}{\sum_{j=1}^{c-1}\exp(\phi_j) + 1}, \quad i = 1, 2, \cdots, c-1, \qquad \widehat{y}_c = 1 - \sum_{i=1}^{c-1} \widehat{y}_i. \qquad (2)$$

The modification amounts to fixing $\exp(\phi_c)$ in the standard SoftMax at 1 eliminating the inherent redundancy of the output weights as also mentioned in [18, p. 150]. The redundancy implies that a particular set of outputs, $\widehat{y}_i$, $i = 1, 2, \cdots, c$ induces a one-dimensional sub-manifold in weight space. The network architecture is shown in Fig. 1.

TRAINING AND REGULARIZATION

Assume that we have a training set $\mathcal{T}$ of N_t related input-output pairs $\mathcal{T} = \{(\boldsymbol{x}(k), \boldsymbol{y}(k))\}_{k=1}^{N_t}$ where

$$y_i(k) = \begin{cases} 1 & \text{if } \boldsymbol{x}(k) \in \mathcal{C}_i \\ 0 & \text{otherwise} \end{cases} \qquad (3)$$

[1]That is, each misclassification is equally serious corresponding to minimal probability of misclassification.

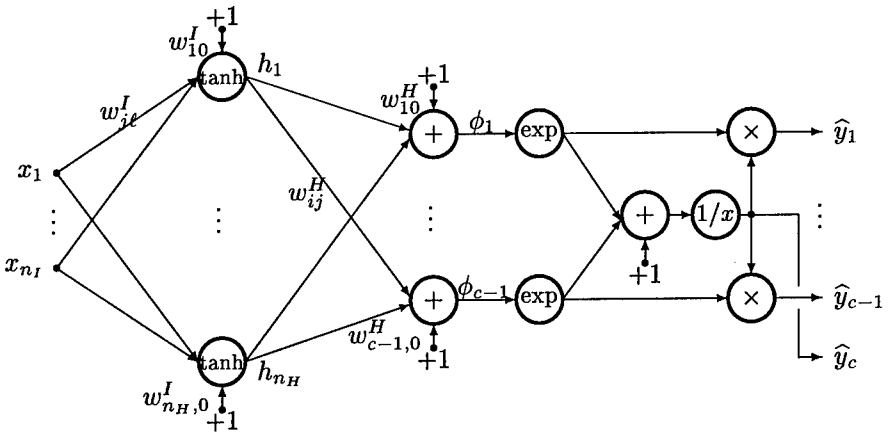


Figure 1: Neural network architecture.

The likelihood of the network parameters is given by (see e.g., [1], [8]),

$$p(\mathcal{T}|\boldsymbol{w}) = \prod_{k=1}^{N_t} p(\boldsymbol{y}(k)|\boldsymbol{x}(k),\boldsymbol{w}) = \prod_{k=1}^{N_t}\prod_{i=1}^{c} (\widehat{y}_i(k))^{y_i(k)} \tag{4}$$

where $\widehat{\boldsymbol{y}}(k) = \widehat{\boldsymbol{y}}(\boldsymbol{x}(k),\boldsymbol{w})$ is a function of the input and weight vectors. The training error is the normalized negative log-likelihood

$$S_{\mathcal{T}}(\boldsymbol{w}) = -\frac{1}{N_t}\log p(\mathcal{T}|\boldsymbol{w}) \equiv \frac{1}{N_t}\sum_{k=1}^{N_t} \ell\left(\boldsymbol{y}(k),\widehat{\boldsymbol{y}}(k);\boldsymbol{w}\right) \tag{5}$$

with $\ell(\cdot)$ denoting the loss given by

$$\ell\left(\boldsymbol{y}(k),\widehat{\boldsymbol{y}}(k);\boldsymbol{w}\right) = \sum_{i=1}^{c} y_i(k)\log\left(1+\sum_{j=1}^{c-1}\exp(\phi_j(\boldsymbol{x}(k)))\right) - \sum_{i=1}^{c-1} y_i(k)\phi_i(\boldsymbol{x}(k)). \tag{6}$$

The objective of training is minimization of the regularized cost function[2]

$$C(\boldsymbol{w}) = S_{\mathcal{T}}(\boldsymbol{w}) + R(\boldsymbol{w},\boldsymbol{\kappa}) \tag{7}$$

where the regularization term $R(\boldsymbol{w},\boldsymbol{\kappa})$ is parameterized by a set of regularization parameters $\boldsymbol{\kappa}$. Training provides the estimated weight vector $\widehat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} C(\boldsymbol{w})$ and is done using a Gauss-Newton scheme,

$$\boldsymbol{w}^{\text{new}} = \boldsymbol{w}^{\text{old}} - \eta\cdot\boldsymbol{J}^{-1}(\boldsymbol{w}^{\text{old}})\boldsymbol{\nabla}(\boldsymbol{w}^{\text{old}}) \tag{8}$$

[2]This might be viewed as a maximum a posteriori (MAP) method.

where η is the step-size (line search parameter). For that purpose we require the gradient, $\boldsymbol{\nabla}(\boldsymbol{w}) = \partial C/\partial \boldsymbol{w}$, and the Hessian, $\boldsymbol{J}(\boldsymbol{w}) = \partial^2 C/\boldsymbol{w}\boldsymbol{w}^\top$ of the cost function given by,

$$\frac{\partial C}{\partial \boldsymbol{w}}(\boldsymbol{w}) = -\frac{1}{N_t}\sum_{k=1}^{N_t}\sum_{i=1}^{c-1}[y_i(k) - \widehat{y}_i(k)]\frac{\partial \phi_i(\boldsymbol{x}(k))}{\partial \boldsymbol{w}} + \frac{\partial R(\boldsymbol{w},\boldsymbol{\kappa})}{\partial \boldsymbol{w}}, \tag{9}$$

$$\boldsymbol{J}(\boldsymbol{w}) = \frac{1}{N_t}\sum_{k=1}^{N_t}\sum_{i=1}^{c-1}\sum_{j=1}^{c-1}\widehat{y}_i(k)\,[\delta_{ij} - \widehat{y}_j(k)]\frac{\partial \phi_i(\boldsymbol{x}(k))}{\partial \boldsymbol{w}}\frac{\partial \phi_j(\boldsymbol{x}(k))}{\partial \boldsymbol{w}^\top} + \frac{\partial^2 R(\boldsymbol{w},\boldsymbol{\kappa})}{\partial \boldsymbol{w}\partial \boldsymbol{w}^\top}. \tag{10}$$

Here δ_{ij} is the Kronecker delta and we have used the Gauss-Newton approximation to the Hessian.

ADAPTING REGULARIZATION PARAMETERS

The available data set, $\mathcal{D}$, of N examples is split into two disjoint sets: a validation set, $\mathcal{V}$, with $N_v = \lceil \gamma N \rceil$ examples for architecture selection and estimation of regularization, and a training set, $\mathcal{T}$, with $N_t = N - N_v$ examples for estimation of network parameters. γ is referred to as the split-ratio.

The validation error of the trained network is given by

$$S_{\mathcal{V}}(\widehat{\boldsymbol{w}}) = \frac{1}{N_v}\sum_{k=1}^{N_v}\ell\,(\boldsymbol{y}(k), \widehat{\boldsymbol{y}}(k); \widehat{\boldsymbol{w}}) \tag{11}$$

where the sum runs over the N_v validation examples. $S_{\mathcal{V}}(\widehat{\boldsymbol{w}})$ is thus an estimate of the generalization error defined as the expected loss: $G(\widehat{\boldsymbol{w}}) = E_{\boldsymbol{x},\boldsymbol{y}}\{\ell(\boldsymbol{y}, \widehat{\boldsymbol{y}}; \widehat{\boldsymbol{w}})\}$, where $E_{\boldsymbol{x},\boldsymbol{y}}\{\cdot\}$ denotes the expectation w.r.t. the joint input-output distribution.

Aiming at adapting the regularization parameters $\boldsymbol{\kappa}$ so that the validation error is minimized we can apply the iterative scheme suggested in [9]:

$$\boldsymbol{\kappa}^{\text{new}} = \boldsymbol{\kappa}^{\text{old}} - \mu\frac{\partial S_{\mathcal{V}}}{\partial \boldsymbol{\kappa}}(\widehat{\boldsymbol{w}}(\boldsymbol{\kappa}^{\text{old}})) \tag{12}$$

where μ is a step-size and $\widehat{\boldsymbol{w}}(\boldsymbol{\kappa}^{\text{old}})$ is the estimated weight vector using the regularization parameter $\boldsymbol{\kappa}^{\text{old}}$. Suppose the regularization term is linear in the regularization parameters, i.e.,

$$R(\boldsymbol{w},\boldsymbol{\kappa}) = \boldsymbol{\kappa}^\top \boldsymbol{r}(\boldsymbol{w}) = \sum_{i=1}^{q}\kappa_i r_i(\boldsymbol{w}) \tag{13}$$

where κ_i are the regularization parameters and $r_i(\boldsymbol{w})$ the associated regularization functions. The gradient of the validation error then equals [9]:

$$\frac{\partial S_{\mathcal{V}}}{\partial \boldsymbol{\kappa}}(\widehat{\boldsymbol{w}}) = -\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{w}^\top}(\widehat{\boldsymbol{w}})\cdot \boldsymbol{J}^{-1}(\widehat{\boldsymbol{w}})\cdot \frac{\partial S_{\mathcal{V}}}{\partial \boldsymbol{w}}(\widehat{\boldsymbol{w}}). \tag{14}$$

Consider the specific case of weight decay regularization with separate weight decays for input-to-hidden and hidden-to output layers, i.e.,

$$R(\boldsymbol{w}, \boldsymbol{\kappa}) = \kappa_I \cdot |\boldsymbol{w}^I|^2 + \kappa_H \cdot |\boldsymbol{w}^H|^2 \tag{15}$$

where $\boldsymbol{\kappa} = [\kappa_I, \kappa_H]^\top$ and $\boldsymbol{w} = [\boldsymbol{w}^I, \boldsymbol{w}^H]$ with $\dim(\boldsymbol{w}^I) = m_I$, $\dim(\boldsymbol{w}^H) = m_H$ and $\dim(\boldsymbol{w}) = m = m_I + m_H$.

The gradient then yields,

$$\frac{\partial S_{\mathcal{V}}}{\partial \kappa_I}(\widehat{\boldsymbol{w}}) = -2(\widehat{\boldsymbol{w}}^I)^\top \cdot \boldsymbol{g}_{m_I}, \quad \frac{\partial S_{\mathcal{V}}}{\partial \kappa_H}(\widehat{\boldsymbol{w}}) = -2(\widehat{\boldsymbol{w}}^H)^\top \cdot \boldsymbol{g}_{m_H} \tag{16}$$

where $\boldsymbol{g} = [\boldsymbol{g}_{m_I}, \boldsymbol{g}_{m_H}] = \boldsymbol{J}^{-1}(\widehat{\boldsymbol{w}}) \cdot \partial S_{\mathcal{V}}(\widehat{\boldsymbol{w}})/\partial \boldsymbol{w}$.

In summary the algorithm for adapting regularization parameters consists of the following 8 steps:

1. Choose the split ratio γ between training and validation set sizes.
2. Initialize $\boldsymbol{\kappa}$ and the weights of the network.
3. Train the network with fixed $\boldsymbol{\kappa}$ to achieve $\widehat{\boldsymbol{w}}(\boldsymbol{\kappa})$. Calculate the validation error $S_{\mathcal{V}}$.
4. Calculate the gradient $\partial S_{\mathcal{V}}/\partial \boldsymbol{\kappa}$ cf. Eq. (14). Initialize the step-size μ.
5. Update $\boldsymbol{\kappa}$ using Eq. (12).
6. Retrain the network from the previous weights and calculate the validation error $S_{\mathcal{V}}$.
7. If no decrease in validation error then perform a bisection of μ and goto step 5; otherwise, continue.
8. Repeat steps 4–7 until the relative change in validation error is below a small percentage or, e.g., the 2-norm of the gradient $\partial S_{\mathcal{V}}/\partial \boldsymbol{\kappa}$ is below a small number.

PRUNING

In order to reduce and optimize the network architecture we suggest to use a pruning scheme, e.g., Optimal Brain Damage (OBD) [11]. An alternative method is Optimal Brain Surgeon (OBS) [5]; however, in a series of experiments we noticed that extreme care is essential in order not to underestimate the saliencies [16]. Thus OBS is less robust than OBD.

OBD ranks the weights according to importance or *saliency*. Here we use the validation error based OBD proposed in [9]. The saliency for weight i is given by

$$\varrho_i = -\widehat{w}_i \frac{\partial S_{\mathcal{V}}}{\partial w_i}(\widehat{\boldsymbol{w}}) + \frac{1}{2}\widehat{w}_i^2 \frac{\partial^2 S_{\mathcal{V}}}{\partial w_i^2}(\widehat{\boldsymbol{w}}). \tag{17}$$

By repeatedly removing weights with small saliencies and retraining the resulting network, a nested family of network architectures is obtained. The

validation error (or an alternative measure of generalization performance[3]) is then used for selecting the optimal architecture.

EXPERIMENTS

We test the performance of the adaptive regularization algorithm on a vowel classification problem. The data are based on the Peterson and Barney database [17]. The classes are vowel sounds characterized by the first four formant frequencies. 76 persons (33 male, 28 female and 15 children) have pronounced $c = 10$ different vowels (IY IH EH AE AH AA AO UH UW ER) two times. This results in a data base of totally 1520 examples. The database is the verified database described in [22] where all data[4] are used, including examples where utterance failed of unanimous identification in the listening test (26 listeners). All examples were included to make the task more difficult.

The examples were split into a data set, $\mathcal{D}$, consisting of $N = 760$ examples (16 male, 14 female and 8 children) and an independent test set of the remaining 760 examples. The regularization was adapted by splitting the data set $\mathcal{D}$ equally into a validation set of $N_v = 380$ examples and a training set of $N_t = 380$ examples (8 male, 7 female and 4 children in each set).

Suppose that the network weights are given by $\boldsymbol{w} = [\boldsymbol{w}^I, \boldsymbol{w}^I_{\text{bias}}, \boldsymbol{w}^H, \boldsymbol{w}^H_{\text{bias}}]$ where $\boldsymbol{w}^I$, $\boldsymbol{w}^H$ are input-to-hidden and hidden-to-output weights, respectively, and the bias weights are assembled in $\boldsymbol{w}^I_{\text{bias}}$ and $\boldsymbol{w}^H_{\text{bias}}$. In this example, we use the following weight decay regularization term:

$$R(\boldsymbol{w}, \boldsymbol{\kappa}) = \kappa^I \cdot |\boldsymbol{w}^I|^2 + \kappa^I_{\text{bias}} \cdot |\boldsymbol{w}^I_{\text{bias}}|^2 + \kappa^H \cdot |\boldsymbol{w}^H|^2 + \kappa^H_{\text{bias}} \cdot |\boldsymbol{w}^H_{\text{bias}}|^2. \quad (18)$$

where $\boldsymbol{\kappa} = [\kappa^I, \kappa^I_{\text{bias}}, \kappa^H, \kappa^H_{\text{bias}}]$. We further define the normalized weight decays as $\boldsymbol{\alpha} \equiv \boldsymbol{\kappa} \cdot N_t$. The simulation set-up was:

- Network: 4 inputs, 5 hidden neurons, 9 outputs[5].
- The training input data were normalized to zero mean and unit variance in order to facilitate training and weight initialization.
- Weights were initialized uniformly over $[-0.5, 0.5]$, regularization parameters were initialized at zero. 10 steps in a gradient descent training algorithm (see e.g., [12]) was performed and the weight decays, $\boldsymbol{\kappa}$, were re-initialized at $\lambda_{\text{max}}/10^2$, where λ_{max} is the max. eigenvalue of the Hessian matrix of the cost function. This initialization scheme is motivated by the following observations:
  - Weight decays should be so small that they do not reduce the approximation capabilities of the network significantly.

[3] E.g., the previously suggested algebraic estimate [8], [15].

[4] The database can be retrieved from `ftp://eivind.imm.dtu.dk/dist/data/vowel/PetersonBarney.tar.Z`

[5] We only need 9 outputs since the posterior class probability of the 10th class is given by $1 - \sum_{j=1}^{9} p(C_j|\boldsymbol{x})$.

	NNet	KNN ($K = 9$)
Training	0.105 ± 0.008	0.150
Validation	0.115 ± 0.005	0.158
Test	0.122 ± 0.005	0.199
Test after retrain.	0.119 ± 0.003	0.153

Table 1: Probability of misclassification, *pmc*. For the neural network the averages and standard deviations over 6 runs are reported.

  - They should be so large that the algorithm is prevented from being trapped in a local optimum and numerical instabilities are eliminated.
- Training is now done using a Gauss-Newton algorithm (see e.g., [12]). The Hessian is inverted using the Moore-Penrose pseudo inverse (see e.g., [19]) ensuring that the eigenvalue spread[6] is less than 10^8.
- The regularization step-size η is initialized at 1.
- When the adaptive regularization scheme has terminated we prune 3% of the weights using a validation set based version of the Optimal Brain Damage recipe [9], [11].
- We alternate between pruning and adaptive regularization until the validation error has reached a minimum.
- Finally, remaining weights are retrained on all data using the optimized weight decay parameters.

Table 1 reports the average and standard deviations of the probability of misclassification (*pmc*) over 6 runs for pruned networks using the optimal regularization parameters. Note that retraining on the full data set decreases the test *pmc* slightly on the average; improvement was found in 4 out of 6 runs. For comparison we used a K-nearest-neighbor (KNN) classification, see e.g., [1] and found that $K = 9$ was optimal on the validation set. Note that the neural network performed significantly better. Contrasting the obtained results to other work is difficult. In [20] results on the Peterson-Barney vowel problem are reported, but their data are not exactly the same; only the first 2 formant frequencies were used. Furthermore, different test sets have been used for the different methods presented. The best result reported [13] is obtained by using KNN and reach $pmc = 0.186$ which is somewhat higher than our results.

In Fig. 2 the evolution of the adaptive regularization as well as the pruning algorithm is demonstrated.

CONCLUSIONS

This paper presented a framework for design of neural classifiers which include architecture optimization by pruning and adaptation of regularization

[6]Eigenvalue spread should not be larger than the square root of the machine precision [3].

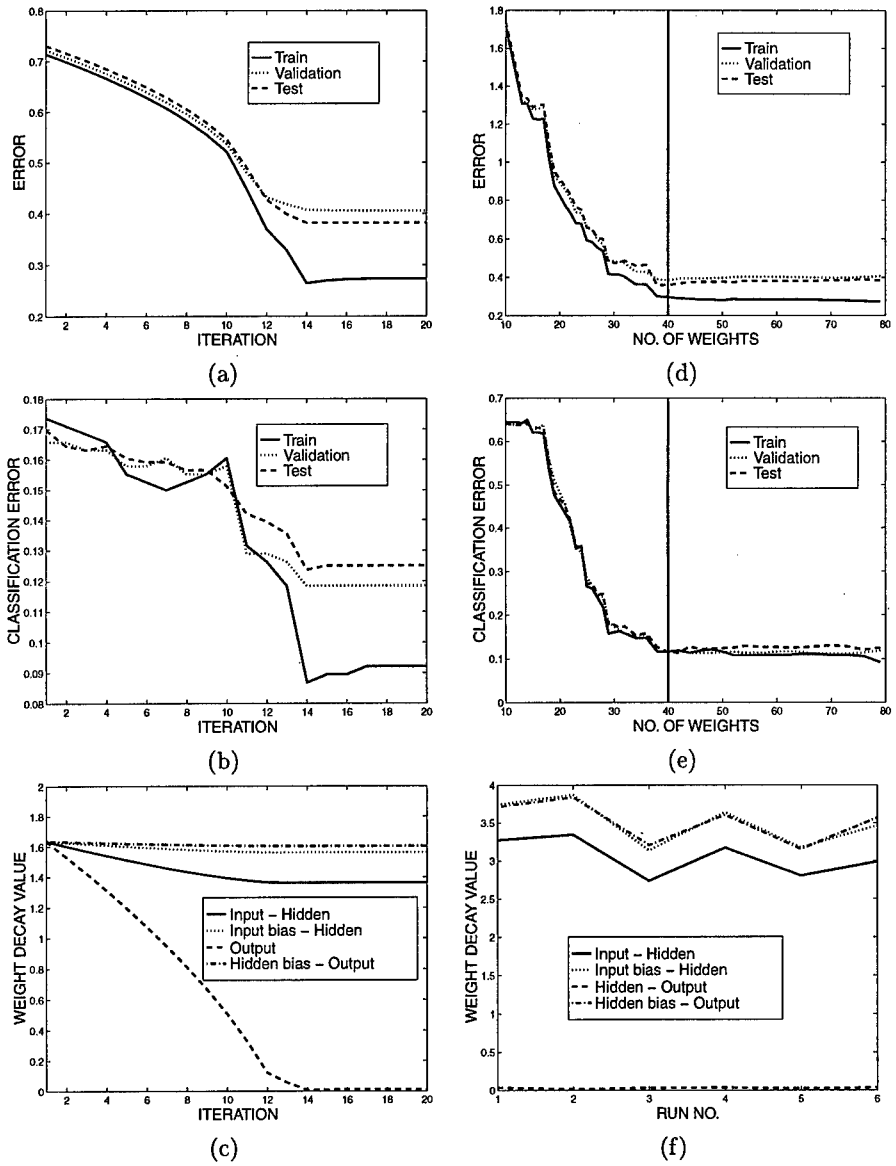


Figure 2: Panels (a), (b) and (c) show the evolution of the adaptive regularization algorithm in a typical run. Optimal weight decays are found by minimizing the validation error in (a). Note that also the test errors decreases. This tendency is also evident in (b) displaying *pmc* even though a small increase is noticed. In (c) the normalized weight decays, $\alpha = \kappa \cdot N_t$, are depicted. (d) and (e) show the evolution of errors and *pmc* during pruning. The optimal network having minimal validation error is indicated by the vertical line. There is only a marginal effect of pruning. Finally, the variation of the optimal normalized weight decays (before pruning) in different runs is shown in (f) and is seen to be relatively small.

parameters. Moreover, an improved neural net architecture was presented. Numerical examples demonstrated the potential of the framework.

ACKNOWLEDGMENTS

This research was supported by the Danish Natural Science and Technical Research Councils through the Computational Neural Network Center (CONNECT). JL furthermore acknowledge the Radio Parts Foundation for financial support. Allan René Rasmussen is thanked for programming assistance.

REFERENCES

[1] C.M. Bishop: **Neural Networks for Pattern Recognition**, Oxford, UK: Oxford University Press, 1995.

[2] J.S. Bridle: "Probabilistic Interpretation of Feedforward Classification Network Outputs with Relationships to Statistical Pattern Recognition," **Neurocomputing - Algorithms, Architectures and Applications**, Berlin Germany: Springer-Verlag, vol. 6, pp. 227–236, 1990.

[3] J.E. Dennis & R.B. Schnabel: **Numerical Methods for Unconstrained Optimization and Non-linear Equations**, Englewood Cliffs, New Jersey: Prentice-Hall, 1983.

[4] S. Geman, E. Bienenstock & R. Doursat, "Neural Networks and the Bias/Variance Dilemma," **Neural Computation**, vol. 4, pp. 1–58, 1992.

[5] B. Hassibi & D.G. Stork: "Second Order Derivatives for Network Pruning: Optimal Brain Surgeon," in S.J. Hanson, J. Cowan & C. Giles (eds.) **Advances in Neural Information Processing Systems 5, Proceedings of the 1992 Conference**, San Mateo, California: Morgan Kaufmann Publishers, 1993, pp. 164–171.

[6] K. Hornik: "Approximation Capabilities of Multilayer Feedforward Networks," **Neural Networks**, vol. 4, pp. 251–257, 1991.

[7] M. Hintz-Madsen, L.K. Hansen, J. Larsen, E. Olesen & K.T. Drzewiecki: "Design and Evaluation of Neural Classifiers - Application to Skin Lesion Classification," in F. Girosi, J. Makhoul, E. Manolakos & E. Wilson (eds.), **Proceedings of the IEEE Workshop on Neural Networks for Signal Processing V**, Piscataway, New Jersey: IEEE, 1995, pp. 484–493.

[8] M. Hintz-Madsen, M. With Pedersen, L.K. Hansen, & J. Larsen: "Design and Evaluation of Neural Classifiers," in S. Usui, Y. Tohkura, S. Katagiri & E. Wilson (eds.), **Proceedings of the IEEE Workshop on Neural Networks for Signal Processing VI**, Piscataway, New Jersey: IEEE, 1996, pp. 223–232.

[9] J. Larsen, L.K. Hansen, C. Svarer & M. Ohlsson: "Design and Regularization of Neural Networks: The Optimal Use of a Validation Set," in S. Usui, Y. Tohkura, S. Katagiri & E. Wilson (eds.), **Proceedings of**

the IEEE Workshop on Neural Networks for Signal Processing VI, Piscataway, New Jersey: IEEE, 1996, pp. 62–71.

[10] J. Larsen, C. Svarer, L. Nonboe Andersen & L.K. Hansen: "Adaptive Regularization in Neural Network Modelling," preprint IMM, DTU, submitted 1997. Available by `ftp://eivind.imm.dtu.dk/dist/1997/larsen.bot.ps.Z`.

[11] Y. Le Cun, J.S. Denker & S.A. Solla: "Optimal Brain Damage," in D.S. Touretzky (ed.) **Advances in Neural Information Processing Systems 2, Proceedings of the 1989 Conference**, San Mateo, California: Morgan Kaufmann Publishers, 1990, pp. 598–605.

[12] L. Ljung: **System Identification: Theory for the User**, Englewood Cliffs, New Jersey: Prentice-Hall, 1987.

[13] D. Lowe: "Adaptive Radial Basis Function Nonlinearities and the Problem of Generalisation," **Proceedings of IEE Conference on Artificial Neural Networks**, 1989, pp. 171–175.

[14] D.J.C. MacKay: "A Practical Bayesian Framework for Backprop Networks," **Neural Computation**, vol. 4, no. 3, pp. 448–472, 1992.

[15] N. Murata, S. Yoshizawaand & S. Amari: "Network Information Criterion — Determining the Number of Hidden Units for an Artificial Neural Network Model," **IEEE Transactions on Neural Networks**, vol. 5, no. 6, pp. 865–872, Nov. 1994.

[16] M.W. Pedersen, L.K. Hansen & J. Larsen: "Pruning with Generalization Based Weight Saliencies: γOBD, γOBS," in D.S. Touretzky, M.C. Mozer & M.E. Hasselmo (eds.) **Advances in Neural Information Processing Systems 8, Proceedings of the 1995 Conference**, Cambridge, Massachusetts: MIT Press, pp. 521–528, 1996.

[17] G.E. Peterson & H.L. Barney: "Control Methods Used in a Study of the Wowels," **JASA** vol. 24, pp. 175–184, 1952.

[18] B.D. Ripley: **Pattern Recognition and Neural Networks**, Cambridge, UK: Cambridge University Press, 1996.

[19] G.A.F. Seber & C.J. Wild: **Nonlinear Regression**, New York, New York: John Wiley & Sons, 1989.

[20] R.S. Shadafan & M. Niranjan: "A Dynamic Neural Network Architecture by Sequential Partitioning of the Input Space," **Neural Computation**, vol. 6, no. 6, pp. 1202–1222, 1994.

[21] C. Svarer, L.K. Hansen & J. Larsen: "On Design and Evaluation of Tapped-Delay Neural Architectures," in **Proceedings of the IEEE International Conference on Neural Networks**, San Francisco, California, USA, vol. 1, pp. 46–51, 1993.

[22] R.L. Watrous: "Current Status of PetersonBarney Vowel Formant Data," **JASA**, vol. 89 pp. 2459–2460, 1991.

Remembering the Past: The Role of Embedded Memory in Recurrent Neural Network Architectures

C. Lee Giles [1,4], Tsungnan Lin [1,2], Bill G. Horne [3]

[1] NEC Research Institute, 4 Independence Way, Princeton, NJ 08540
[2] Department of Electrical Engineering, Princeton University, Princeton, NJ 08540
[3] AADM Consulting, 9 Pace Farm Rd., Califon, NJ 07830
[4] UMIACS, University of Maryland, College Park, MD 20742

Abstract

There has been much interest in learning long-term temporal dependencies with neural networks. Adequately learning such long-term information can be useful in many problems in signal processing, control and prediction.

A class of recurrent neural networks (RNNs), NARX neural networks, were shown to perform much better than other recurrent neural networks when learning simple long-term dependency problems. The intuitive explanation is that the output memories of a NARX network can be manifested as jump-ahead connections in the time-unfolded network.

Here we show that similar improvements in learning long-term dependencies can be achieved with other classes of recurrent neural network architectures simply by increasing the order of the embedded memory. Experiments with locally recurrent networks, and NARX (output feedback) networks show that all of these classes of network architectures can have a significant improvement on learning long-term dependencies as the orders of embedded memory are increased, other things be held constant. These results can be important to a user comfortable with a specific recurrent neural network architecture because simply increasing the embedding memory order of that architecture will make it more robust to the problem of long-term dependency learning.

1 Introduction

Recurrent Neural Networks (RNNs), though capable of representing arbitrary nonlinear dynamical systems [25] and computationally quite powerful [26], can sometimes have difficulty learning even simple temporal behavior. Part of this difficulty

0-7803-4256-9/97/$10.00

has been attributed to the problem of *long-term dependencies* [2, 19], i.e. those problems for which the desired output of a system at time T depends on inputs presented at times $t \ll T$.

In particular Bengio *et al.* [2] showed that if a system is to latch information robustly, then the fraction of the gradient in a gradient-based training algorithm due to information n time steps in the past approaches zero as n becomes large. This effect is called the problem of *vanishing gradient*. Bengio *et al.* claimed that the problem of a vanishing gradient is the essential reason why gradient-descent methods are not sufficiently powerful to learn long-term dependencies.

Several approaches have been suggested to circumvent the problem of vanishing gradients in training RNNs: presetting initial weights by using prior knowledge [6, 9], alternative optimization methods instead of gradient-based [2], reduced description of data [19, 23, 24], architectures that operate on multiple time scales [10, 11] and architectures with high-order gating units[12].

A class of recurrent neural networks called NARX networks can perform much better at learning long-term dependencies when using a gradient descent training algorithm [17]. The intuitive explanation for this behavior is that the output memories of a NARX neural network are manifested as jump-ahead connections in the time-unfolded network that is often associated with algorithms as Backpropagation Through Time (BPTT). These jump-ahead connections provide shorter paths for propagating gradient information, thus reducing the sensitivity of the network to long-term dependencies.

We hypothesize that the similar improvement on learning long-term dependencies can be achieved in other classes of recurrent neural network architectures by increasing the orders of embedded memory. (One of the first uses of embedded memory in recurrent network architectures was that of Jordan [14].) In this paper, we empirically justify this hypothesis by showing the relationship between memory order of a RNN and its sensitivity to long-term dependencies. In Section 2, we discuss three classes of conventional recurrent neural networks architectures: globally recurrent networks (the architecture, not the training procedure, used by Elman) [5]; locally recurrent networks (in particular the Frasconi, Gori and Soda's model) [7]; NARX networks [3, 21], and their corresponding models with a high order embedded memory. In Section 3, we provide a empirical comparison of these architectures by investigating their performance on learning two simple long-term dependencies problems: the latching problem and a grammatical inference problem. These simulations show that these classes of recurrent neural network architectures all demonstrate significant improvement on learning long-term dependencies when the embedded memory order is increased and weights remain relatively the same. Thus, a user of one of these recurrent architectures can readily improve their robustness to long-term memory problems simply by increasing the amount of embedded memory, all other variables remaining constant.

2 Embedding memory order in recurrent neural network architectures

Several recurrent neural network architectures have been proposed; for a collection of papers on the variety see [8]. One taxometric classification for these architectures can be based on the observability of their states: specifically they can be broadly divided into two groups depending on whether or not the states of the network are observable or not [13]. For another taxometric approach based on memory types, see Mozer [20]. For this study we picked three classes of networks: globally recurrent (GR) networks [5], locally recurrent networks (LR) [7], and NARX networks [3, 21]; and their corresponding architectures with high-order embedded memory. It should be pointed out that our embedded memory simply consists of simple tapped delayed values to various neurons and not more sophisticated embedded memory structures [20, 4]. NARX networks are a typical model of networks with observable states. GR networks are a popular class of network with globally connected hidden states, and LR networks belong to locally recurrent network architecture class also with hidden states.

2.1 Globally connected RNNs

These networks (which we will call GR networks) are a class of recurrent networks in which the feedback connections come from the state vector to the hidden layer, as illustrated in Figure 1 (a). These hidden states are sometimes called *context units* in the literature. Suppose such a network with n_u input nodes, n_h hidden nodes of, and n_y output nodes, the dynamic equation can be described by:

$$o_i(t) = f\left(\sum_{j=1}^{n_h} w_{ij}^h o_j(t-1) + \sum_{k=1}^{n_u} w_{ik}^u u_k(t) + w_i^b\right). \tag{1}$$

$$y_i(t) = f\left(\sum_{j=1}^{n_h} w_{ij}^y o_j(t) + w_i^b\right), \tag{2}$$

where $o(t)$ and $y(t)$ denotes the real valued outputs of the hidden and output neurons at time t, and f is the nonlinear function.

This network with a high order of embedded memory differs from standard globally connected recurrent network in that they have more than one state vector per feedback loop. Specially, for a GR network with embedded memory of order m, the dynamic equations of hidden nodes become:

$$o_i(t) = f\left(\sum_{k=1}^{m}\sum_{j=1}^{n_h} w_{ijm}^h o_j(t-k) + \sum_{k=1}^{n_u} w_{ik}^u u_k(t) + w_i^b\right). \tag{3}$$

Figure 1 (b) illustrates an GR network with embedded memory of order two.

2.2 Locally recurrent networks

In this class of networks, the feedback connections are only allowed from neurons to themselves, and the nodes are connected together in a feed forward architecture [1,

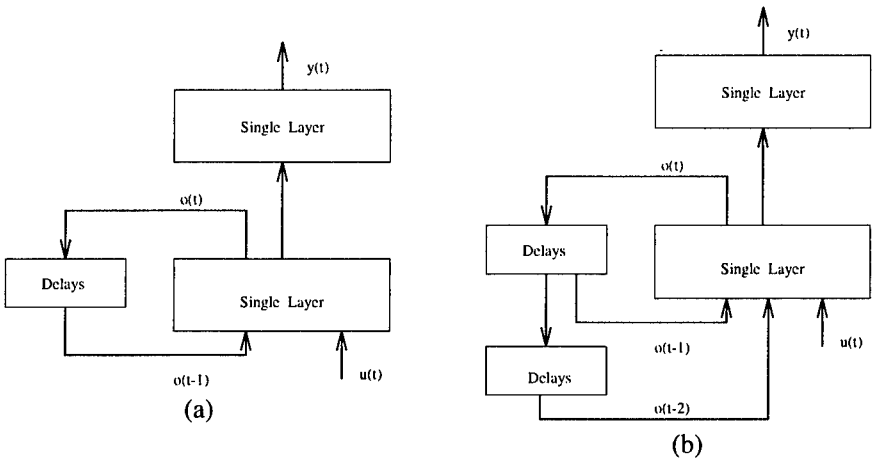


Figure 1: (a) A standard GR network. (b) A GR network with embedded memory of order two.

15, 7, 22, 29]. Specifically, we consider networks proposed by Frasconi *et al.* [7] (we will call LR), as shown in Figure 2 (a). The dynamic neurons of LR networks can be described by

$$o_i(t) = f\left(w_{ii}^h o_i(t-1) + \sum_j w_{ij}^u u_j(t) + w_i^b\right), \tag{4}$$

where $o_i(t)$ denotes the output of the i^{th} node at time t, and f is the nonlinearity. For a network with embedded memory of order m, the output of the dynamic neurons becomes

$$o_i(t) = f\left(\sum_{n=1}^{m} w_{ii}^h o_i(t-n) + \sum_j w_{ij}^u u_j(t) + w_i^b\right). \tag{5}$$

Figure 2 (b) shows a LR network with embedded memory of order two. Locally recurrent models usually differ in where and how much output feedback is permitted; see [29] for a discussion of architectural differences.

2.3 NARX recurrent neural networks

An important class of discrete–time nonlinear systems is the *Nonlinear AutoRegressive with eXogeneous inputs* (NARX) model [3, 18, 27, 28]:

$$y(t) = f\left(u(t-D_u), \ldots, u(t), y(t-D_y), \ldots, y(t-1)\right), \tag{6}$$

where $u(t)$ and $y(t)$ represent input and output of the network at time t, D_u and D_y are the input-memory and output-memory order, and the function f is a nonlinear function. When the function f can be approximated by a Multilayer Perceptron, the resulting system is called a *NARX recurrent neural network* [3, 21].

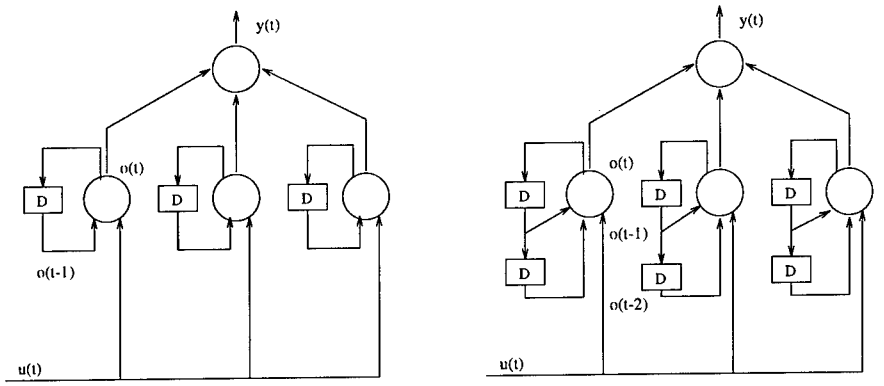


Figure 2: (a) A standard LR network. (b) A LR network with embedded memory of order two.

In this paper, we shall consider NARX networks with zero input order. Thus, the operation of the network id defined by

$$y(t) = f\left(u(t), y(t - D_y), \ldots, y(t-1)\right). \tag{7}$$

Figure 3 shows a NARX architecture with output memory of order 3.

3 Experimental Results

Simulations were performed to explore the effect of embedded memory on learning long-term dependencies in these three different recurrent network architectures. The long-term dependency problems investigated were the latching problem and a grammatical inference problem. These problems were chosen because they are simple and should be easy to learn but exemplify the long-term dependency issue. For more complex problems involving long-term dependencies see [12].

In order to establish some metric for comparison of the experimental results, we gave the recurrent networks sufficient resources (number of weights and training examples, adequate training time) to readily solve the problem but held the the number of weights approximately invariant across all architectures. Also note that in some cases the order of the embedded memory is the same.

3.1 The latching problem

This experiment evaluates the performance of different recurrent network architectures with various order of embedded memory on a problem already used for studying the difficulty in learning long-term dependencies [2, 11, 17].

This problem is a minimal task designed as a test that must necessarily be passed in order for a network to robustly latch information [2]. In this two-class problem, the class of a sequence depends only on the first 3 time steps, the remaining values

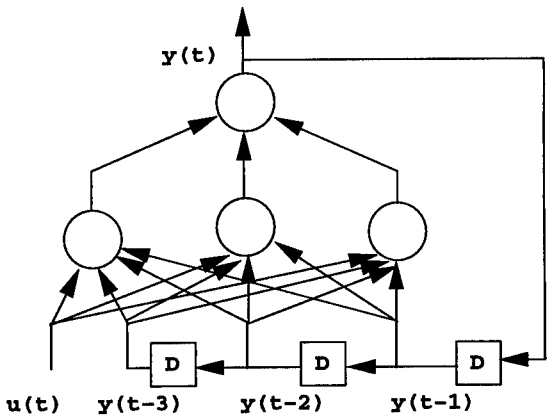


Figure 3: A NARX network with output memory of order 3.

Architecture	*Network Description*				*# weights*
	Memory order	# states	# hidden neurons	In-hid-out	
GR(1)	1	6	6 nodes	3-6-1	85
GR(2)	2	10	5 nodes	3-5-1	91
GR(3)	3	12	4 nodes	3-4-1	81
NARX(2)	2	2	11 nodes	3-11-1	111
NARX(4)	4	4	8 nodes	3-8-1	97
NARX(6)	6	6	6 nodes	3-6-1	85
LR(1)	1	14	14 nodes	3-14-1	109
LR(2)	2	22	11 nodes	3-11-1	110
LR(3)	3	27	9 nodes	3-9-1	111

Table 1: Architecture description of different recurrent networks used for the latching problem. We used the hyperbolic tangent function as the nonlinear function for each neuron.

in the sequence is uniform noise. There are three inputs $u_1(t)$, $u_2(t)$, and a noise input $e(t)$. Both $u_1(t)$ and $u_2(t)$ are zero for all times $t > 1$. At time $t = 1$, $u_1(1) = 1$ and $u_2(1) = 0$ for samples from class 1, and $u_1(1) = 0$ and $u_2(1) = 1$ for samples from class 2. The class information of each strings is contained in $u_1(t)$ and $u_2(t)$. We used two delay elements for both $u_1(t)$ and $u_2(t)$ in order to hold the class information until $t = 3$. The noise input $e(t)$ is given by

$$e(t) = \begin{cases} 0 & t \leq 3 \\ U(-b, b) & 3 < t \leq T \end{cases} \tag{8}$$

where $U(-b, b)$ are samples drawn uniformly from $[-0.155, 0.155]$. Target information was only provided at the end of each sequence. For comparison, our training particulars are identical to those of [2]. For strings from class one, a target value of 0.8 was chosen, for class two, -0.8 was chosen. The length of the noisy sequence could be varied in order to control the span of long-term dependencies. For our experiment, the input sequences were 1 and 0 and were one-hot encoded into two input neurons with trainable weights.

For each of these three architectures previously discussed, several networks with different orders of embedded memory were trained. To compare the effects of different orders of embedded memory in every class of networks on learning long-term dependencies while holding as many other factors as possible constant, particular attention was paid to equalize the number of weights. Table 1 gives a detailed description of all networks used in the latching problem. The weight connected the noisy input was fixed as 1.0. In order to learn the task, the networks have to develop two attractors to latch the information and still remain inside the basin of the attractors of being resistant to noise when $t > 3$. The ability of learning this minimal problem is a measure of the effectiveness of propagating the gradient for different neural network architectures with various memory orders.

The length of noisy inputs, T, was varied from 10 to 60 in increments of 2. For each value of T, we ran 50 simulations. For each simulation, 30 strings were generated from each class and the initial weights were randomly distributed in the range $[-0.5, 0.5]$.

The network was trained with a MSE cost function using simple BPTT algorithm with a learning rate of 0.1 for a maximum of 200 epochs. Updates occurred at the end of each string and the error was back-propagated the full length of the string. If the absolute error between the output of the network and the target value was less than 0.6 on all strings, the simulation was terminated and determined successful. If the simulation exceeded 200 epochs and did not correctly classify all strings, then the simulation was ruled a failure.

Figures 4 (a) to (c) show plots of the percentage of those runs that were successful for different classes of networks with different orders of embedded memory. It is clear from these plots that the network architectures with high order embedded memory become increasingly less sensitive to long–term dependencies as the memory order was increased.

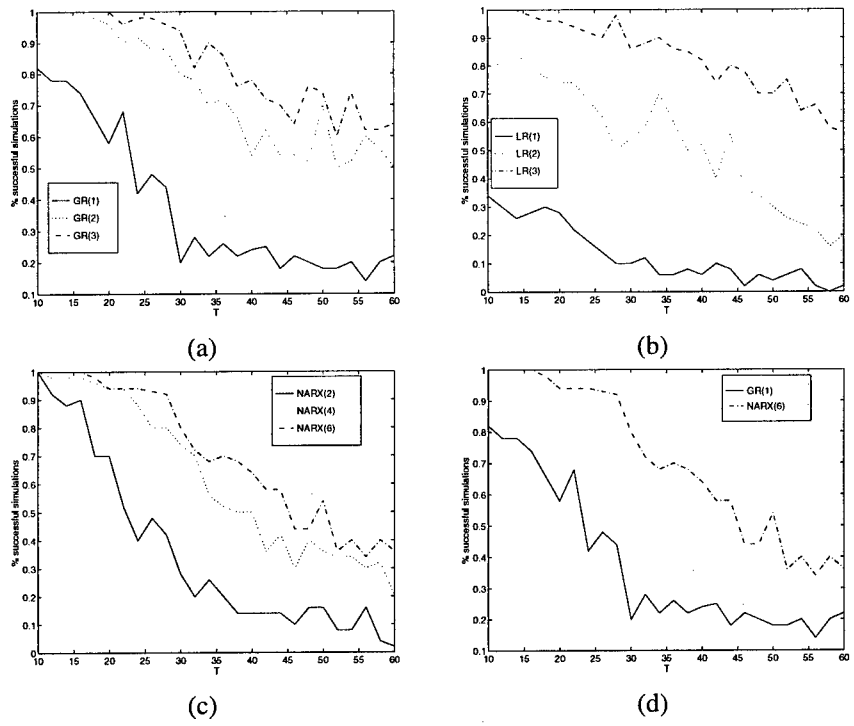


Figure 4: Plots of percentage of successful simulations on the latching problem from 50 runs as a function of T, the length of input strings, for different recurrent network architectures with different orders of embedded memory: (a) Globally connected RNN (GR) , (b) Locally connected RNN (LR), (c) NARX, (d) NARX v.s. GR(1).

An interesting comparison between the architectures GR(1) and NARX(6) is shown in Figure 4 (d). Since the two architectures have the exact same number of weights, hidden nodes, and states, the only difference is the amount of memory order.NARX networks perform better than the GR networks at learning the latching problem.

3.2 Grammatical Inference (Tree Automata) Problem

In previous problem, the inputs to the network were followed by a noise term. In this experiment, we consider learning to classify strings of boolean values, which are labelled according to some prespecified automata.

In this example, the class of a string is completely determined by its input symbol at some prespecified time t. For instance, Figure 5 shows a five-state automaton used in the experiments, in which the class of each string is determined by the third input symbol. When that symbol is "1", the string is accepted; otherwise, it is rejected.

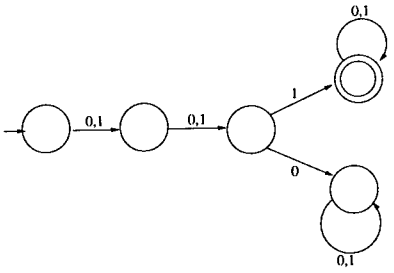


Figure 5: A five-state tree automaton. The unlabeled arrow is the start state and the double circled state is the the acceptance state.

By increasing the length of the strings to be learned, we will be able to control the span of long-term dependencies, in which the output will depend on input values far in the past.

Again, we noted the same improvement on learning long-term dependencies obtained by increasing the order of embedded memory in each class of recurrent neural network architectures. For more details regarding the experiment, please see [16].

4 Conclusion

Motivated by the analysis of the problem of learning long-term dependencies and the success of NARX networks on problems including grammatical inference and nonlinear system identification [13], we explore the ability of other recurrent neural networks with a high order of embedded memory on problems that involve long-term dependencies. We chose three classes of recurrent neural network architectures based on state-observerability: hidden state globally recurrent and locally recurrent networks, and observeable state NARX networks.

We tested this approach of extending memory in conventional recurrent neural networks on two simple long-term dependency problems. Our experimental results show that each of these classes of recurrent neural networks architectures can demonstrate significant improvement on learning long-term dependencies when the memory order of the network is increased.

The intuitive explanation for this behavior is that the embedded memories are manifested as jump–ahead connections in the unfolded network that is often used to describe algorithms like Backpropagation Through Time. These jump-ahead connections provide a shorter path for propagating gradient information, thus reducing the sensitivity of the network to long-term dependencies. Another explanation is that the states do not necessarily need to propagate through nonlinearities at every time step, which may avoid a degradation in gradient due to the partial derivative of the nonlinearity. We speculate that using increased memory order will also help other recurrent network architectures on learning long-term dependency problems.

References

[1] A.D. Back and A.C. Tsoi. FIR and IIR synapses, a new neural network architecture for time series modeling. *Neural Computation*, 3(3):337–350, 1991.

[2] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[3] S. Chen, S.A. Billings, and P.M. Grant. Non–linear system identification using neural networks. *International Journal of Control*, 51(6):1191–1214, 1990.

[4] B. de Vries and J. C. Principe. The gamma model — A new neural model for temporal processing. *Neural Networks*, 5:565–576, 1992.

[5] J.L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.

[6] P. Frasconi, M. Gori, M. Maggini, and G. Soda. Unified integration of explicit rules and learning by example in recurrent networks. *IEEE Transactions on Knowledge and Data Engineering*, 7(2):340–346, 1995.

[7] P. Frasconi, M. Gori, and G. Soda. Local feedback multilayered networks. *Neural Computation*, 4:120–130, 1992.

[8] C.L. Giles, G.M. Kuhn, and R.J. Williams. Dynamic recurrent neural networks: Theory and applications. *IEEE Transactions on Neural Networks*, 5(2), 1994. Special Issue.

[9] C.L. Giles and C.W. Omlin. Inserting rules into recurrent neural networks. In S.Y. Kung, F. Fallside, J. Aa. Sorenson, and C.A. Kamm, editors, *Neural Networks for Signal Processing II, Proceedings of The 1992 IEEE Workshop*, pages 13–22, Piscataway, NJ, 1992. IEEE Press.

[10] M. Gori, M. Maggini, and G. Soda. Scheduling of modular architectures for inductive inference of regular grammars. In *ECAI'94 Workshop on Combining Symbolic and Connectionist Processing, Amsterdam*, pages 78–87. Wiley, August 1994.

[11] S. El Hihi and Y. Bengio. Hierarchical recurrent neural networks for long-term dependencies. In *Advances in Neural Information Processing Systems 8*. MIT Press, Cambridge, MA, 1996.

[12] S. Hochreiter and J. Schmidhuber. Long short term memory. Technical Report FKI-207-95, Fakultat fur Informatik, Technische Universitat Munchen, Munchen, 1995.

[13] B.G. Horne and C.L. Giles. An experimental comparison of recurrent neural networks. In *Advances in Neural Information Processing Systems 7*, pages 697–704. MIT Press, 1995.

[14] M. I. Jordan. Attractor dynamics and parallelism in a connectionist sequential machine. In *Proceedings of the Eighth Conference of the Cognitive Science Society*, pages 531–546. Erlbaum, 1986.

[15] Steve Lawrence, Andrew Back, A.C. Tsoi, and C. Lee Giles. The gamma MLP – using multiple temporal resolutions for improved classification. In *Neural Networks for Signal Processing VII – Proceedings of the 1997 IEEE Workshop*. Florida, 1997.

[16] T. Lin, B.G. Horne, and C.L. Giles. How embedded memory in recurrent neural network architectures helps learning long-term temporal dependencies. Technical Report UMIACS-TR-96-28 and CS-TR-3626, Institute for Advanced Computer Studies, University of Maryland.

[17] T. Lin, B.G. Horne, P. Tino, and C.L. Giles. Learning long-term dependencies in narx recurrent neural networks. *IEEE Transactions on Neural Networks*, 7(6):1329–1338, 1996.

[18] L. Ljung. *System identification : Theory for the user*. Prentice-Hall, Englewood Cliffs, NJ, 1987.

[19] M. C. Mozer. Induction of multiscale temporal structure. In J.E. Moody, S. J. Hanson, and R.P. Lippmann, editors, *Neural Information Processing Systems 4*, pages 275–282. Morgan Kaufmann, 1992.

[20] Michael C. Mozer. Neural net architectures for temporal sequence processing. In A.S. Weigend and N.A. Gershenfeld, editors, *Time Series Prediction*, pages 243–264. Addison–Wesley, 1994.

[21] K.S. Narendra and K. Parthasarathy. Identification and control of dynamical systems using neural networks. *IEEE Trans. on Neural Networks*, 1(1):4, 1990.

[22] P.S. Sastry, G. Santharam, and K.P. Unnikrishnan. Memory neuron networks for identification and control of dynamical systems. *IEEE Transactions on Neural Networks*, 5(2):306–319, 1994.

[23] J. Schmidhuber. Learning complex, extended sequences using the principle of history compression. *Neural Computation*, 4(2):234–242, 1992.

[24] J. Schmidhuber. Learning unambiguous reduced sequence descriptions. In J. E. Moody, S. J. Hanson, and R. P. Lippman, editors, *Advances in Neural Information Processing Systems 4*, pages 291–298. San Mateo, CA: Morgan Kaufmann, 1992.

[25] D.R. Seidl and R.D. Lorenz. A structure by which a recurrent neural network can approximate a nonlinear dynamic system. In *Proceedings of the International Joint Conference on Neural Networks 1991*, volume II, pages 709–714, July 1991.

[26] H.T. Siegelmann and E.D. Sontag. On the computational power of neural nets. *Journal of Computer and System Sciences*, 50(1):132–150, 1995.

[27] H.-T. Su and T.J. McAvoy. Identification of chemical processes using recurrent networks. In *Proceedings of the American Controls Conference*, volume 3, pages 2314–2319, 1991.

[28] H.-T. Su, T.J. McAvoy, and P. Werbos. Long–term predictions of chemical processes using recurrent neural networks: A parallel training approach. *Industrial Engineering and Chemical Research*, 31:1338–1352, 1992.

[29] A.C. Tsoi and A. Back. Locally recurrent globally feedforward networks, a critical review of architectures. *IEEE Transactions on Neural Networks*, 5(2):229–239, 1994.

LOW SENSITIVITY TIME DELAY NEURAL NETWORKS WITH CASCADE FORM STRUCTURE

Andrew D. Back[1],Bill G. Horne[2], Ah Chung Tsoi[3], C. Lee Giles[4]
[1]Brain Information Processing Group, Frontier Research Program,
The Institute of Physical and Chemical Research (RIKEN),
2-1 Hirosawa, Wako-shi, Saitama 351-01 Japan
[2]AADM Consulting, 9 Pace Farm Rd, Califon, NJ 07830 USA
[3]Faculty of Informatics, University of Wollongong
Northfields Avenue, Wollongong, Australia
[4]NEC Research Institute, 4 Independence Way, Princeton, NJ 08540. USA

Abstract

In current practice, tapped delay line models such as the time delay neural network (TDNN) are commonly implemented using a *direct form* structure. In this paper, we show that the problem of high parameter sensitivity, well known in linear systems, also applies to nonlinear models such as the TDNN. To overcome the consequent numerical problems, we propose a *cascade form* TDNN (CTDNN) and show its advantages over the commonly used direct form TDNN.

1 Introduction

In signal processing and control applications, there is much interest in nonlinear adaptive filters based on neural networks. The tapped delay line has been employed in [10] to modify the classic multilayer perceptron (MLP) for signal processing. It is easily observed[1] that the tapped delay line can be considered as a finite impulse response (FIR) filter [4, 18]:

$$y(t) = \sum_{i=0}^{m} q^{-i} b_i x(t) \tag{1}$$

[1]Note that we use notation which includes both time and the q-operator, as is standard practice in the literature.

0-7803-4256-9/97/$10.00

where b_i are constants and $q^{-1}x(t) \triangleq x(t-1)$. Eqn (1) can be equivalently written as

$$y(t) = \prod_{i=1}^{m}(1 - c_i q^{-1})x(t) \tag{2}$$

If c_i is a complex number, then it will appear with its complex conjugate for real $y(t)$.

This model has been proven to be capable of universally approximating a functional [7, 16]. However, for real world applications, there are a number of other aspects which also need to be considered when implementing cost effective models in hardware. One area normally considered, is the round-off error and quantization error due to the finite arithmetic wordlengths.

It is known in the digital signal processing literature [14, 19] that with the realization of FIR and IIR (infinite impulse response) filters using devices capable of only finite precision arithmetic, (1) is more sensitive to round-off effects and coefficient quantization than (2). Eqn (1) is known as a direct form implementation, while (2) is known as a cascade form [19].

While the sensitivity properties of linear filters are well known, in general, it appears that this area has not been considered in neural networks. It appears to be quite common that experiments are conducted using time delay neural networks with a direct form structure and not to consider potential problems of numerical effects such as parameter sensitivity. Hence, in this paper we investigate the sensitivity of TDNN networks which are based on (1) and propose a new class of TDNN architecture based on the cascade form model. The main aim of the paper, is therefore to demonstrate the validity and usefulness of using cascade form structures in time delay neural networks.

2 A Cascade Form Time Delay Neural Network

As implied in (1), we may equivalently formulate the input layer of the TDNN[2] as FIR filters $G_{0j}(q)$ from the input to the jth unit in the first hidden layer.

This approach allows us to easily consider various extensions to the basic TDNN structure. Hence we may have

$$G_0(q) = \sum_{i=0}^{n_b} b_i q^{-i} \tag{3}$$

[2]For convenience we consider a single-input single-output model, with a tapped delay line occurring only at the input layer, though further extensions could be derived.

Eqn (3) may be written equivalently as a cascade form structure as follows

$$G_0(q) = \prod_j^M B_{0j}(q) \tag{4}$$

where typically, $\{B_0(q)\}$ is a first or second order section of the form $(1-c_iq^{-1})$, where c_i may be real or complex; if c_i is complex, then it must occur in a complex conjugate pair for a real output signal. This structure is a specialized form of the TDNN, which we term a cascade form time delay neural network (CTDNN). Hence we term the model represented by (3) a direct form TDNN (DTDNN). The results can be applied equally well to the full TDNN structure, but for clarity we consider this simpler model form.

3 Parameter Sensitivity Analysis of Time Delay Neural Networks

In neural networks, the issue of sensitivity to errors in the weights has typically been approached in a probabilistic framework [2, 8, 9, 17]. Here, the approach used is to extend the usual method of parameter sensitivity analysis used in linear systems based on the model poles and zeros, to nonlinear systems. Although the use of poles and zeros in nonlinear systems may be thought by some to be inappropriate, this is not the case as shown in [5, 6] (see also [15]).

The method presented in [6] is based on approximating a particular class of nonlinear system described by

$$y(t) = F\left(G(q)x(t)\right) \tag{5}$$

where $F(\cdot)$ is a memoryless nonlinearity and for $G(q) = B_0(q)$, (5). In this case, (5) is a special case of the TDNN. If $G(q)$ is a SIMO (single-input multiple-output) structure, i.e. a parallel filter bank, and $F(\cdot)$ is a MISO nonlinear map, then the more general form of TDNN is obtained. For clarity of presentation, but without loss of generality, we consider the case where $G(q) = B_0(q)$ and $F(\cdot)$ is SISO.

We may approximate $F(\cdot)$ by a power series expansion, giving

$$\begin{aligned} y(t) &= \sum_{k=0}^{n} \bar{\beta}_{0k} + g_1(x(t)) + \ldots + g_p(x^p(t)) \\ &\quad + O\left(x^{i_0}(t)x^{i_1}(t-1)\ldots x^{i_n}(t-n)\right) + \ldots \end{aligned} \tag{6}$$

where $\{\bar{\beta}\}$ is the set of parameters obtained from the approximation of $F(\cdot)$ and

$$g_j(x^j(t)) = \Psi_j(q)x^j(t) \tag{7}$$

$$\Psi_j(q) = \sum_{k=0}^{n} \bar{\beta}_{jk} q^{-k} \tag{8}$$

Hence the TDNN model can be approximated by a summation of subsystems which employ linear transfer functions $\Psi_j(q)$ as shown in (8). The parameter sensitivity measure S_{ij} normally used for linear systems [12], can be applied to the subsystems which form the approximate TDNN model and consequently to the TDNN. The parameter sensitivity measure is defined as

$$\begin{aligned} S_{ij} &= \frac{d\lambda_i}{db_j} \\ &= \frac{\lambda_i^{n-j}}{\prod_{k \neq i}(\lambda_i - \lambda_k)} \end{aligned} \tag{9}$$

where S_{ij} is the sensitivity of the ith root λ_i, with respect to the parameter b_j of the polynomial $B(q)$. A high parameter sensitivity implies that a small change in a parameter will lead to a large change in the model behaviour, as determined by the roots of the polynomial of the FIR filter [1,3]. Hence problems of round-off error and coefficient quantization may become significant. So also, errors can be introduced into adaptive models as the weights are updated, but the model behaviour differs from that required by the update.
Therefore the following observations can be made:

Observation 1. A DTDNN model is subject to possible problems of high parameter sensitivity.

Observation 2. If a method can be given which reduces the sensitivity of a linear model, then the same technique will reduce the parameter sensitivity of the subsystems within the approximate TDNN model and hence, of the TDNN model itself.

We may define the parameter sensitivity of a DTDNN in terms of the sensitivities of the subsystems within the approximate TDNN model.

Definition 1 *The sensitivity S_N of a DTDNN with M tapped delay inputs and N_0 input units is given by*

$$S_N \triangleq \max_i (S_i : i = 1, ..., N_0) \tag{10}$$

$$S_i \triangleq \max_j (S_{ij} : j = 1, ..., M) \tag{11}$$

Definition 2 *The sensitivity S_{Nc} of a CTDNN with $(M+1)/2$ (M odd) or $(M+2)/2$ (M even) first or second order filter sections going to each of N_0*

input units is given by

$$S_{Nc} \triangleq \max_i (S_{ci} : i = 1, ..., N_0) \quad (12)$$

$$S_{ci} \triangleq \max_j (S_{cij} : j = (M+1)/2 \ (M \ odd), (M+2)/2 \ (M \ even)) \quad (13)$$

Note that S_{ij} is the sensitivity of an Mth order direct form filter, while S_{cij} is the sensitivity of a 2nd order filter. Hence we obtain the following theorem.

Theorem 1 *The sensitivity S_N of a direct form TDNN is less than or equal to the sensitivity S_{Nc} of a cascade form TDNN .*

Proof. From (9), we have, for $j = n$, the last stage in an n tap delay line:

$$S_{in} = \frac{1}{\prod_{k \neq i}(\lambda_i - \lambda_k)} \quad (14)$$

It is sufficient to consider two cases:

1. $|\lambda_i - \lambda_k| \gg 0$ In this case, S_{ij} will be small, which implies that the difference between the sensitivities for cascade form and direct form models will be of negligible consequence.

2. $0 \approx |\lambda_i - \lambda_k| \ll 1$ In this case, for an arbitrarily large tapped delay order n,

$$\max(S_{ij}) \geq \lim_{k \to \infty} \frac{1}{\prod_k |\lambda_i - \lambda_k|} \quad (15)$$

$$= \infty \quad (16)$$

For a CTDNN, $n = 2$, it is evident that

$$S_{Nc} \leq S_N \quad (17)$$

Hence, when it matters most, i.e. for systems exhibiting high sensitivity as in case 2, the CTDNN will provide better sensitivity properties than an equivalent DTDNN. This effect will in general, be greater as the order of the tapped delay line increases.

4 Examples

To better elucidate the difference between the networks[3], we will give an example to illustrate the possible performance differences between CTDNN and

[3] We would like to point out however, that while the problems considered here are artificially constructed, they are meant to serve as an example of possible behaviour. For

DTDNN models. We consider a system identification problem[4] where we have a system to be identified $\mathcal{S}$ and two models $\mathcal{M}_d$ and $\mathcal{M}_c$ which correspond to a DTDNN and a CTDNN respectively. In order to illustrate the phenomenon, we assume a perfect representation in each case, i.e., no training is involved. We consider two models and the effect of coefficient quantization on the accuracy of the models in each case.

We consider the difference between the DTDNN and a CTDNN when subjected to finite wordlength (FWL) restrictions. In this experiment, we consider a TDNN with 10 tap delays and 1 hidden unit The corresponding polynomial is given by

$$\begin{aligned} G(q) &= 1.000 - 4.200q^{-1} + 8.280q^{-2} - 10.524q^{-3} + 10.091q^{-4} - 8.090q^{-5} \\ &\quad +5.629q^{-6} - 3.212q^{-7} + 1.329q^{-8} - 0.338q^{-9} + 0.039q^{-10} \end{aligned} \tag{18}$$

For each of the direct form and cascade form models, wordlengths of 16 and 8 bits were tested. The method used to round off the weights is

$$\tilde{\theta} = 2^{-w}\text{round}\,(2^w\theta) \tag{19}$$

where θ is the original weight, $\tilde{\theta}$ is the FWL weight and w is the word length in bits.

The behaviour of the models are compared by examining the shift in the position of the zeros (roots) of the FIR filter polynomials at the input of the TDNNs and comparing them to the position of the true zeros. These zeros are plotted in Fig 1, where it can be observed the conventional direct form TDNN indicates significant movement in the zeros. The cascade form model however, performs very well and does not appear to suffer significantly from the reduced precision in the weights.

Note that the zeros are not shifted by much in the regions where the sensitivity is low. The regions of high sensitivity (as expected from the sensitivity analysis - see (9) and Theorem 1), are where the zeros have large magnitudes and are close together, i.e. for real valued systems, this is near the (1,0) point.

It has also been observed in other experiments (not shown here due to lack of space) that the direct form model sometimes obtains nonminimum phase zeros. For system identification and control applications this can cause problems [11].

some applications, there may be negligible difference between the architectures, however we propose that the choice is up to the user. If one wishes to avoid the potential problems indicated in this paper, then we propose that the cascade form or parallel form TDNNs are preferable to the direct form TDNN. If one is confident that the application will not present a problem for direct form structures, then the direct form may be used. It is noted that except in speech synthesis, direct forms are infrequently used other than in second order sections [13]. This is possible because in speech synthesis the poles of the system function are widely separated [14].

[4]Due to lack of space, only one experiment will be shown in this section, however other experiments indicate the same type of problem.

Clearly, even in simple TDNNs, the accuracy degrades rapidly when using a smaller number of bits in the weights. For real-world applications, e.g. automotive or other consumer-oriented products, this issue could be of concern where longer wordlengths add undesirable cost to the product. On the other hand, in areas where a higher precision computing resources are available, the problem may be viewed in a slightly different manner. Here, for any given wordlength, when modelling low frequency systems, the cascade TDNN offers a potentially higher accuracy than the direct form structure.

It should of course be noted that the use of cascade form filters is widespread in digital signal processing, and as such, the neural network structure proposed here should come as no surprise to those familiar with such methods. However, in view of the widespread usage of the classical TDNN (direct form) structure, we felt that it may be worth drawing to the attention of the neural network community, the potential advantages of some slight modifications to the structure of the TDNN. Moreover, the advantages of cascade form structures would also carry over to multilayer structures. In related work, we have shown that alternative discrete-time operators can be used to produce lower sensitivity structures in both feedforward and recurrent networks [5,6].

We do not consider the learning problem in this paper, however, based on the numerical improvements to the architecture, we would expect that there should also be enhanced performance in on-line learning. This would apply particularly, of course, in FWL modelling of high sensitivity systems. In any training algorithms, the advantages would accrue from using the cascade structure directly, as opposed to computing the poynomial weights on a highly accurate computer, transforming them and then 'downloading' them to the cascade form model, although such an approach could be used if required for some special purpose.

5 Conclusions

Current neural network models which process spatial or temporal data typically use direct form substructures. In this paper we have proposed a new class of model which generalizes the classical TDNN structure to allow an input structure which consists of synapses such as cascade, parallel or cascade-parallel filters instead of the conventional direct form structure.

The advantage of this approach is particularly evident in terms of reducing sensitivity to quantization errors in the network weights. The resulting network will have a higher accuracy than the conventional direct form network for a given finite word length implementation. This is most evident when the system being modelled has low frequency characteristics.

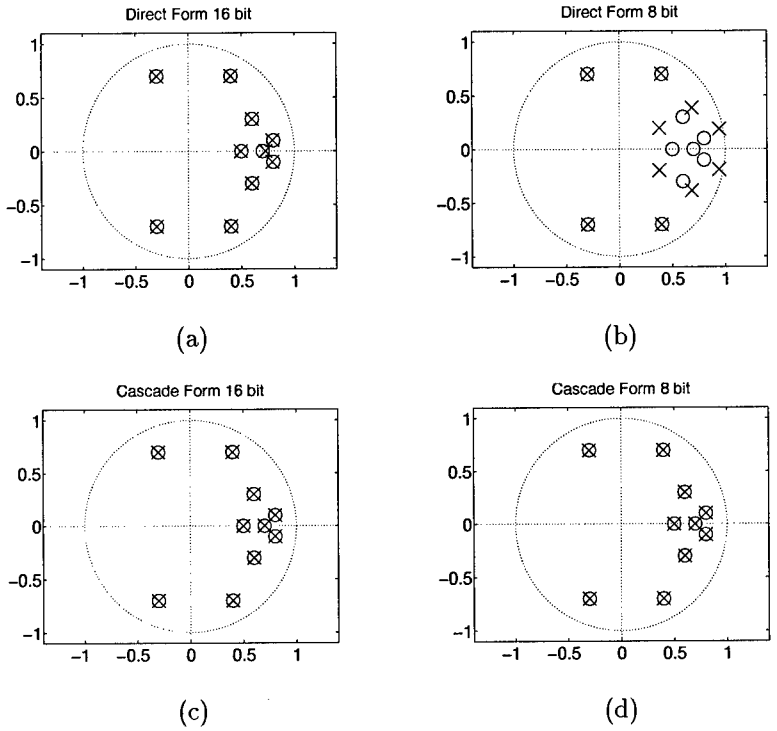


Figure 1: The zeros for the polynomials are shown for (a) 16 bit direct form, (b) 8 bit direct form, (c) 16 bit cascade form, (d) 8 bit cascade form models. In each case, the true zeros are shown for reference. To distinguish the true zeros and the model zeros, 'x' symbols are used to indicate the model zeros. Here it can be noted that the cascade model is almost identical to the true system, while the direct form is significantly different. The cascade model gives significant advantages over the direct form model. The main differences occur in the low frequency region however, indicating problems in modelling low frequency data.

Acknowledgements

The first author acknowledges support from the Australian Research Council and the Frontier Research Program, RIKEN, Japan. The second author acknowledges partial support from the Australian Research Council.

References

[1] R.C. Agarwal and C.S. Burrus. New recursive digital filter structures having very low sensitivity and roundoff noise. *IEEE Trans. Circuits, Syst.*, CAS-22(12):921–927, 1975.

[2] C. Alippi, V. Piuri, and M. Sami. Sensitivity to errors in artificial neural networks: a behavioural approach. *IEEE Trans. Circuits, Syst. I: Fundamental Theory and Applications*, 42(6):358–361, 1995.

[3] K.J. Astrom, P. Hagander, and J. Sternby. Zeros of sampled systems. *Automatica*, 20(1):31–38, 1984.

[4] A.D. Back and A.C. Tsoi. FIR and IIR synapses, a new neural network architecture for time series modelling. *Neural Computation*, 3(3):375–385, 1991.

[5] A.D. Back and A.C. Tsoi. A low sensitivity recurrent neural network. *to appear in Neural Computation*, 1997.

[6] A.D. Back, A.C. Tsoi, B.G Horne, and C.L. Giles. Alternative discrete-time operators and their application to nonlinear models. Technical Report CS-TR-3738 and UMIACS-TR-97-03, Institute for Advanced Computer Studies University of Maryland, College Park, Md 20742, 1997.

[7] T. Chen and H. Chen. Approximations of continuous functionals by neural networks with application to dynamic systems. *IEEE Trans. Neural Networks*, 4(6):910–918, 1993.

[8] J.Y. Choi and C-H. Choi. Sensitivity analysis of multilayer perceptrons with differentiable activation functions. *IEEE Trans. Neural Networks*, 3:101–107, 1992.

[9] G. Dündar and K. Rose. The effects of quantization on multilayer perceptrons. *IEEE Trans. Neural Networks*, 6(6):1446–1451, 1995.

[10] K. Lang, A.H. Waibel, and G.E. Hinton. A time delay neural network architecture for isolated word recognition. *Neural Networks*, 3:23–44, 1990.

[11] L. Ljung and T. Soderstrom. *Theory and Practice of Recursive Identification.* The MIT Press, Cambridge, MA, 1983.

[12] P.E. Mantey. Eigenvalue sensitivity and state-variable selection. *IEEE Trans. Automat. Control*, AC-13(3):263–269, 1968.

[13] A.V. Oppenheim and R.W. Schafer. *Discrete-time Signal Processing.* Prentice-Hall, Englewood Cliffs, NJ, 1989.

[14] L.R. Rabiner and B. Gold. *Theory and Application of Digital Signal Processing.* Prentice-Hall, Englewood Cliffs, NJ, 1975.

[15] W.J. Rugh. *Nonlinear Theory: The Volterra-Wiener Approach.* The Johns Hopkins University Press, Baltimore, Maryland, 1981.

[16] I.W. Sandberg. Approximation theorems for discrete-time systems. *IEEE Trans. Circuits, Syst.*, 38(5):564–566, 1991.

[17] M. Stevenson, R. Winter, and B. Widrow. Sensitivity analysis of feedforward neural networks to weight errors. *IEEE Trans. Neural Networks*, 1:71–90, 1990.

[18] E.A. Wan. Time series prediction by using a connectionist network with internal delay lines. In A.S. Weigend and N.A. Gershenfeld, editors, *Time Series Prediction: Forecasting the Future and Understanding the Past*, volume Proc. Volume XV Santa Fe Institute Studies in the Sciences of Complexity, pages 195–217, Reading,MA, 1994. Addison-Wesley.

[19] D. Williamson. Delay replacement in direct form structures. *IEEE Trans. Acoust., Speech, Signal Processing*, 34(4):453–460, April 1988.

Novel Projection Pursuit Indices for Feature Extraction and Classification: An Inter-comparison in a Remote Sensing Application

Charles M. Bachmann *
Naval Research Laboratory, Remote Sensing Division
Remote Sensing Hydrodynamics Branch, Code 7255
4555 Overlook Ave. SW, Washington, D. C. 20375
email: bachmann@nrl.navy.mil

May 10, 1997

Abstract

Projection Pursuit [7] [10] (PP) techniques are used to search for statistically interesting low-dimensional projections of complex, high-dimensional data. These projections reveal data structure useful for automatic classification applications. We derive a novel class of Projection Pursuit algorithms, comparing them with related PP algorithms [7] [8] [11] [2] [1]. Texture-based cloud detection in Airborne Visible/Infrared Imaging Spectrometer (AVIRIS) imagery from the Jet Propulsion Laboratory is provided as a basis for inter-comparison.

1 Projection Pursuit: Background

In order to identify potentially meaningful data structures in high-dimensional data sets, "Projection Pursuit" [7] (PP) techniques are used to search for statistically interesting low-dimensional projections. PP is an iterative search technique that converges to extrema of a projection index or cost function,

*This work was supported in part by a grant of High Performance Computing (HPC) time from the following Department of Defense HPC Centers: Army Research Laboratory SGI Power Challenge Array, the Maui High Performance Computing Center, and the Naval Research Laboratory SGI Origin 2000.

that measures the degree of multi-modality or departure from normality of the projected data distribution. One of the first PP methods was proposed by Friedman and Tukey [7], who coined the term "Projection Pursuit." Their cost function was the product of two functions, one that measures the spread of the projected data (a trimmed variance to ensure insensitivity to outliers), and another that measures compactness within a particular distance scale. Historically, the Friedman-Tukey cost function can be seen as an innovative step beyond Principal Component Analysis (PCA). As shown in Figure 1, using PCA to find maximal data variance of *all data samples* is not necessarily the most informative for classification. This Figure also emphasizes the fact that an orthonormal decomposition does not always reveal the most relevant information from the perspective of class separation.

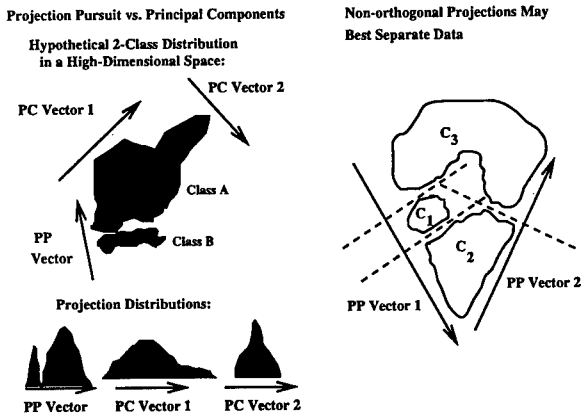


Figure 1: (Left) PP vs. PCA: PCA corresponds to a special case of PP in which the Projection Index is maximal variance (power), which will not always reveal clusters in the data. More sophisticated Projection Indices can be defined to favor the discovery of multi-modal projected distributions. (Right) Three classes of data located in distinct clusters optimally separated by hyperplanes (dashed lines) perpendicular to the projection vectors; hyperplanes correspond to scalar thresholds of projected data; note that optimal PP vectors are not orthogonal.

For a unit projection vector $\hat{w}_k$, input data sample $\vec{f}_i$, and projection:

$$c_k(i) = \hat{w}_k \cdot \vec{f}_i, \tag{1}$$

the Friedman and Tukey PP algorithm is:

$$Maximize: \; I(c_k) = S(c_k)N(c_k) \tag{2}$$

where:

$$S(c_k) = \sqrt{\frac{\sum_{i=1}^{N_p-n}(c_k(i) - E(c_k))^2}{N_p - n}} \quad (3)$$

$$N(c_k) = \sum_{i=1}^{N_p}\sum_{j=1}^{N_p} g(r_k(i,j))\theta(R - r_k(i,j)) \quad (4)$$

$$with\ r_k(i,j) = |\ c_k(i) - c_k(j)\ |, \quad (5)$$

$$g(r_k(i,j)) : Monotone\ Decreasing,\ \&\ \theta : a\ step\ function \quad (6)$$

N_p is the total number of patterns, n is a small fraction of outliers removed at both extremes of the projection, $E(c_k)$ is the mean projection value, and R is a scalar cut-off outside of which pairs of points are excluded in the compactness function, $N(c_k)$. The resulting search algorithm favors the discovery of multi-modal structure in the data. In the absence of the factor, $N(c_k)$, the original PP index would reduce to $S(c_k)$ and would be equivalent to PCA. In [7], the search procedure consisted of optimizing 1D or 2D projections of the data, one at a time. A number of other PP methods were later developed. A fundamental concept in these later algorithms is to find projections that are least normal [8] [10]. Typically, the projection vectors in these methods are optimized serially with each subsequent search vector optimized on a residual after subtraction of structure from the previous projection. In this sense, they differ from unsupervised neural network learning algorithms that also implement a form of PP but do so by jointly optimizing a set of projections (see e.g. [11]). No proof exists to suggest whether serial or joint optimization is superior.

2 A Novel Class of Projection Pursuit Indices

In revisiting the original PP Index designed by Friedman and Tukey, one can see a limitation: the factor measuring the degree of data spread, $S(c_k)$, does not directly focus on the *spread between clusters*, but rather measures the *spread of the whole data set.* One approach to circumventing this difficulty is to define an entirely new projection index that focuses directly on the degree of departure from normality (see for e.g. [8]). Alternatively, our approach is to replace $S(c_k)$ with a function, $D(\tilde{c}_k)$, that directly measures the spread outside a clustering/nearness scale α_k. The symbol $\tilde{c}_k$ refers to the nonlinear data projection that replace the linear projections c_k defined in Equation 1:

$$\tilde{c}_i^{(n)} = \sigma(\sum_j L_{ij}^{(n)} c_j^{(n)}), \quad (7)$$

$$where\ \sigma(x) = a\ \tanh(a\lambda x),\ (a, \lambda\ \text{constant}), \quad (8)$$

$$c_j^{(n)} = \vec{w}_j^{(n-1)} \cdot \vec{\tilde{c}}^{(n-1)} + b_j^{(n)}, \quad (9)$$

and where $\vec{w}_j^{(n-1)}$ is the jth modifiable projection vector, that weights the inputs $\vec{\tilde{c}}^{(n-1)}$ from layer $n-1$, and $b_j^{(n)}$ is the bias. The coupling/constraint matrix $L_{ij}^{(n)}$ is either fixed or modifiable (see the next Section). The nonlinear projections are no longer constrained to be on the unit sphere, and, importantly, are expressed as saturating nonlinearities that remove sensitivity to extreme outliers in the data. In fact, this latter property allows us to retain data points originally ignored by the Friedman-Tukey Index (see (3) and (4)).

We optimize projections jointly as in PP neural network approaches. Our new PP Index is:

$$\Xi = \frac{1}{K}\sum_{k=1}^{K}\Xi_k = \frac{1}{K}\sum_{k=1}^{K} N^{cont.}(\tilde{c}_k) D^{cont.}(\tilde{c}_k) \quad (10)$$

where $N^{cont.}$ and $D^{cont.}$ are given by: [1]

$$N^{cont.}(\tilde{c}_k) = E_{pairs,(\mu,\nu)}[g(\tilde{r}_k(\mu,\nu))] \quad (11)$$

$$D^{cont.}(\tilde{c}_k) = E_{pairs,(\mu,\nu)}[\ \tilde{r}_k^2(\mu,\nu)(1 - g(\tilde{r}_k(\mu,\nu)))\ ] \quad (12)$$

$$with\ \tilde{r}_k^2(l,m) = (\tilde{c}_k(l) - \tilde{c}_k(m))^2,\ \&\ g(\tilde{r}_k(\mu,\nu)) = e^{-(\frac{\tilde{r}_k(\mu,\nu))}{\alpha_k})^2} \quad (13)$$

We have chosen a particular form for the nearness function $g(\tilde{r}_k(\mu,\nu))$ in order to derive specific expressions for a particular case of Equation 10. In our novel PP algorithm, each nearness function $g(\tilde{r}_k(\mu,\nu))$ has a clustering scale factor α_k associated with it. Each α_k is obtained by multiplying an initial estimate of the standard deviation of the projected data by a random number drawn from a user-determined range. The α_k can be modifiable, although for the results in this paper, they were static. Selecting a range of α_k is useful because clusters and other structure may be visible on more than one scale in the data depending on how the high-dimensional data is viewed, i.e. depending on the orientation of the PP search vector in the high-dimensional data space.

In $D^{cont.}(\tilde{c}_k)$, we use the squared distance weighted by the factor $(1 - g(\tilde{r}_k(\mu,\nu)))$, since this product will directly measure inter-point spread of projected data, weighting more heavily those distances outside the nearness/clustering scale α_k. Thus, it will be maximal for well-separated clusters existing within the scale α_k. If the projected distribution is multi-modal, $D^{cont.}(\tilde{c}_k)$ will measure the spread of the modes better than $S(c_k)$, which

[1] Here $E_{pairs,(\mu,\nu)}$ means expected value over sample pairs.

simply measures overall data spread by calculating the variance about the mean of the projected distribution.

Minimization of our PP Index in (10) by gradient descent leads to the following modification equation for the ith projection vector $\vec{w}_i^{(n-1)}$:

$$\begin{aligned}\Delta w_{ij}^{(n-1)} &= -\eta(t)\frac{\partial \Xi^{(n)}}{\partial w_{ij}^{(n-1)}} \\ &= -\eta(t)\frac{1}{K}\sum_{k=1}^{K}[\, N^{cont.}(\tilde{c}_k^{(n)})\frac{\partial D^{cont.}(\tilde{c}_k^{(n)})}{\partial w_{ij}^{(n-1)}} + \\ &\quad \frac{\partial N^{cont.}(\tilde{c}_k^{(n)})}{\partial w_{ij}^{(n-1)}} D^{cont.}(\tilde{c}_k^{(n)})\,] \qquad (14)\end{aligned}$$

where:

$$\frac{\partial N^{cont.}(\tilde{c}_k^{(n)})}{\partial w_{ij}^{(n-1)}} = -E_{pairs,(\mu,\nu)}[\frac{\psi_{kij}}{\alpha_k^2} g(\tilde{r}_k(\mu,\nu))] \qquad (15)$$

$$\begin{aligned}\frac{\partial D^{cont.}(\tilde{c}_k^{(n)})}{\partial w_{ij}^{(n-1)}} &= E_{pairs,(\mu,\nu)}[\ \psi_{kij}(\mu,\nu)\ (\ (\frac{\tilde{r}_k(\mu,\nu)}{\alpha_k})^2 g(\tilde{r}_k(\mu,\nu)) + \\ &\quad (1-g(\tilde{r}_k(\mu,\nu)))\ )\ ] \qquad (16)\end{aligned}$$

$$\begin{aligned}\psi_{kij}(\mu,\nu) &= \frac{1}{2}\frac{\partial \tilde{r}_k^2(\mu,\nu)}{\partial w_{ij}^{(n-1)}} \\ &= (\tilde{c}_k^{(n)}(\mu) - \tilde{c}_k^{(n)}(\nu))\cdot \\ &\quad ((\tilde{c}_k^{(n)}(\mu))'\tilde{c}_j^{(n-1)}(\mu) - (\tilde{c}_k^{(n)}(\nu))'\tilde{c}_j^{(n-1)}(\nu))L_{ki} \qquad (17)\end{aligned}$$

$$(\tilde{c}_i^{(n)})' = \lambda(a-\tilde{c}_i^{(n)})(a+\tilde{c}_i^{(n)}). \qquad (18)$$

2.1 Implementation Details

Because of the pairwise calculations inherent in the quantities, D, N, $\frac{\partial D}{\partial w_{ij}}$, and $\frac{\partial D}{\partial w_{ij}}$, memory and computational requirements in a storage intensive version of the algorithm would be potentially quite severe, growing quadratically with the number of sample estimates used at each update. This potential problem is circumvented by an on-line implementation that uses stochastic gradient descent, with estimates of D, N, $\frac{\partial D}{\partial w_{ij}}$, and $\frac{\partial D}{\partial w_{ij}}$ computed by a running average. For example:

$$N^{cont.}(\tilde{c}_k)(t) = \epsilon\, N^{cont.}(\tilde{c}_k)(t-1) + (1-\epsilon)g(\tilde{r}_k(\tilde{c}_k(t),\tilde{c}_k(t-1))) \qquad (19)$$

By choosing a small and decreasing learning rate, $\eta(t)$, and an appropriate value of ϵ, the distribution of sample pairs can be estimated sufficiently to ensure gradient descent. Typically, we let the learning rate decrease as $\eta(t) = \eta_0/(1 + ln(\frac{t}{\tau_\eta}))$. A theoretical justification for choosing a logarithmically decreasing learning rate can be made using arguments from simulated annealing [9].

Another detail of the implementation is that the coupling matrix $L_{ij}^{(n)}$ may be fixed or modifiable. In the latter case, $L_{ij}^{(n)}$ is modified by gradient ascent to maximize the relative entropy of the projections, $\tilde{c}$: [2]

$$let\ L_{ij}^{(n)}(0) = \begin{pmatrix} 1-\mu, \text{for } i=j \\ -\mu, \text{for } i \neq j \end{pmatrix} \tag{20}$$

$$and\ \zeta = -\frac{1}{N^2}\sum_{k,l}\tilde{q}_k ln(\frac{\tilde{q}_l}{\tilde{q}_k}),\ \ with\ \tilde{q}_k = \frac{1}{2a}(\tilde{c}_k^{(n)} + a), \tag{21}$$

$$\Delta L_{ij}^{(n)} = \eta_L \frac{\partial \zeta}{\partial L_{ij}^{(n)}} \tag{22}$$

$$= \eta_L \frac{1}{N} c_j^{(n)}[\ (\tilde{c}^{(n)})_i'(\ 1 + ln(a + \tilde{c}_i^{(n)}) - E_p(ln(a + \tilde{c}^{(n)}))\ ) - (a - \tilde{c}_i^{(n)}) E_p(a + \tilde{c}_i^{(n)})\ ]. \tag{23}$$

If the scale of the initial coupling matrix, set by μ, is sufficiently larger than the scale used to randomly initialize the weights w_{ij}, and the learning rates for L and $\vec{w}$ are chosen appropriately, then the change in the effective projection vectors $\delta\vec{w}_{effec} = \delta(L \cdot \vec{w}) = L\delta\vec{w} + (\delta L)\vec{w} = L\eta_{\vec{w}}\nabla_{\vec{w}}\Xi + \eta_L(\nabla_L\zeta)\vec{w}$, will be dominated by the PP gradient term with adjustments from the second term dependent on $\nabla_L\zeta$. Our experiments to date suggest that using the modifiable coupling matrix rather than the fixed constraint may accelerate the maximization of the PP Index Ξ. At present we use both forms in experiments, although results reported here were obtained with the modifiable constraint matrix.

2.2 Results for a Remote Sensing Application

To investigate the potential usefulness of PP techniques for the extraction of textural features in future Multi-Angle Imaging Spectro-Radiometer (MISR) [6] data, we have analyzed 17 high resolution images from the Airborne Visible/Infrared Imaging Spectrometer (AVIRIS) [13] operated by the Jet Propulsion Laboratory. We use the four (out of 224) AVIRIS channels centered at the MISR wavelengths of 443, 555, 670, and 865 nm [6]. An example of

[2] $E_p()$ refers to the expected value across projections in the network.

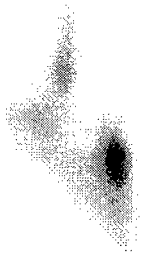


Figure 2: Two-dimensional histogram of data projected onto a pair of projections in our new PP network after training; each axis is the degree of overlap with one of the selected projections. Inputs were gray-level difference vector (GLDV) histograms [5] [15] derived from AVIRIS band 5; input windows were 12x12 pixels.

multi-modal structure found in the AVIRIS data by our new PP algorithm is shown in Figure 2. A typical end result for one of the novel AVIRIS test images using related PP techniques [1] [2] from our earlier research along with standard texture features [14] is illustrated in Figure 3. As Figure 3 demonstrates, the majority of incorrect identifications are at the edges of the clouds where the features are a mixture of cloudy and cloud-free imagery. In Figure 4, results obtained with our new PP algorithm combined with cross-entropy based backward propagation (BPCE) [12] as a back-end classifier are compared against these other approaches. For comparable architecture sizes, the new PP algorithm with BPCE appears to be better than the other PP algorithms individually combined with BPCE. Also, the new PP algorithm with BPCE is statistically closest to the "all" cases that achieved the best performance by combining features derived by all of the other PP algorithms along with standard statistical features [14]. Mean performance on the novel held-out image for the best "all" case is (93.5 $\pm$ 6.8) % cloud pixels detected with false alarm rate of (10.6 $\pm$ 10.0) %. The size of the errors bars in Figure 4 is somewhat large due to several factors: the limited number of trials (eight - ten per paradigm), the fact that there is a high degree of variability within even this limited database, and the complexity of the search space, that is full of local minima. Also, this database is significantly smaller than is necessary to completely characterize class variability and adequately represent the cloud detection problem domain in the training set. A larger statistical sample of experiments and further tuning of the individual PP algorithms is needed to obtain a better estimate of the relative merits of the approaches,

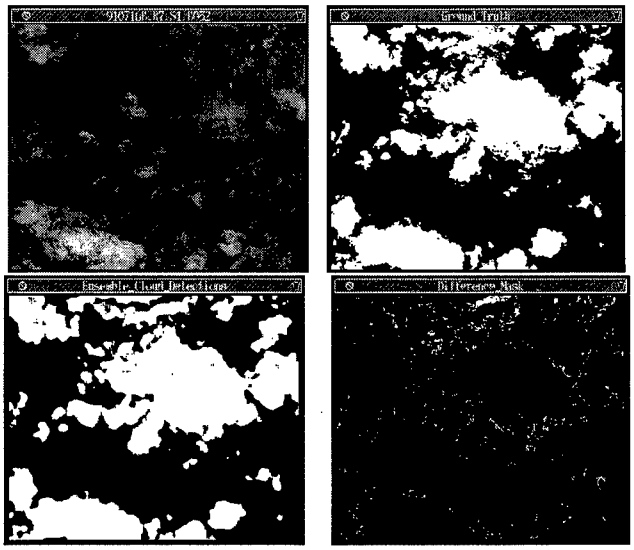


Figure 3: Cloud detection in a novel AVIRIS test scene. (Upper left) Near infra-red band (865nm) for an AVIRIS scene in the test set; band is one of 4 chosen to correspond spectrally to future MISR bands; colormap is black (low) to white (high). (Upper right) Human interpretation of cloud pixel location, white = cloud, black= no cloud. (Lower left) Cloud detection result from an ensemble network using extracted Wavelet Projection Pursuit (WPP) features [2], BCM-PP [4] [11] [1] features from GLDV histograms, BCM-PP features from simply normalized pixel intensities [1], and standard standard statistical moments [14] from GLDV; features in the ensemble model were extracted from all four spectral channels; white = cloud, black = no cloud. (Lower right) difference mask: black = no error, gray = false-alarm, white = false negative; most errors are on cloud/no-cloud boundaries. Cloud detection rate for this novel test image was 94.0 % with false alarm rate 4.8 % .

but these first results are encouraging and suggest that features from the new PP algorithm might be more robust than those found by the other PP algorithms and should be added to the "all" case to look for further improvement in future experiments.

References

[1] C. M. Bachmann, E. E. Clothiaux, J. W. Moore, K. J. Andreano, "An Ensemble Approach to Automatic Cloud Identification in AVIRIS Imagery Using BCM Projection Pursuit," in Neural Networks for Signal Processing IV: Proceedings of the 1994 IEEE Workshop, Sept 6-8, 1994, Ermioni, Greece, pp. 394-403.

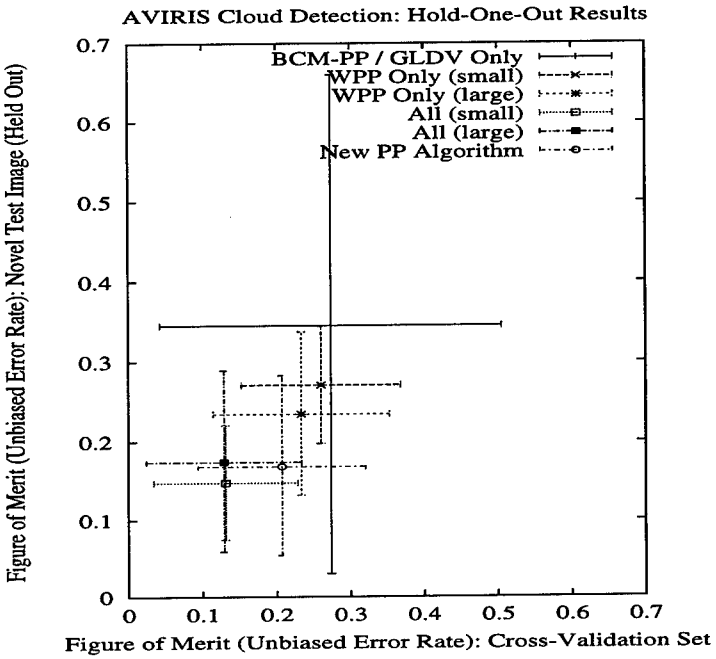


Figure 4: An inter-comparison of cloud detection results over multiple trials: $F_{merit} = \sqrt{\epsilon^2_{false\ alarm} + \epsilon^2_{false\ negative}}$ for the novel held out image vs. the cross validation set; $\epsilon_{false\ alarm}$ is the false alarm rate, $\epsilon_{false\ negative} = 1 - \rho_{detect}$ is the false negative rate, and ρ_{detect} is the rate of detection. F_{merit} (unbiased error) is the distance of network performance from an ideal detector on a receiver operating curve (ROC). Results for the novel PP algorithm combined with BPCE as back-end classifier achieve lower error than the other PP algorithms with BPCE; the new algorithm is statistically closest to the ensemble "all" cases that combine features from all of the other PP algorithms and achieve the best performance. "All" paradigms correspond to BPCE networks receiving input features from Wavelet Projection Pursuit (WPP)[2] projections of albedos, BCM-PP [4] [11] [1] projections of albedos, BCM-PP applied to GLDV histograms [1] , and standard statistical features [14] from GLDV. "All" cases showed statistically significant improvement over any other combination of features with average held-out image performance for the best: (93.5 $\pm$ 6.8) % cloud pixels detected with false alarm rate (10.6 $\pm$ 10.0) %.

[2] C. M. Bachmann, E. E. Clothiaux, D. Q. Luong, "Wavelet Projection Pursuit for Feature Extraction and Cloud Detection in AVIRIS and AVHRR Imagery," in Proceedings of the International Geoscience and Remote Sensing Symposium (IGARSS96'), Lincoln, Nebraska, May 27-31, 1996, Vol. I, pp. 356-359.

[3] C. M. Bachmann, E. E. Clothiaux, D. Q. Luong, "Texture Feature Extraction by Wavelet Projection Pursuit for Cloud Detection in AVIRIS and AVHRR Imagery," in Neural Networks for Signal Processing VI: Proceedings of the 1996 IEEE Workshop, Kyoto, Japan, Sept. 4-6, 1996, pp. 351-360.

[4] E. L. Bienenstock, L. N. Cooper, P. W. Munro, "Theory for the Development of Neuron Selectivity: Orientation Specificity and Binocular Interaction in Visual Cortex," *J. Neuroscience*, Vol. 2, No. 1, pp. 32-48, 1982.

[5] D. W. Chen, S.K. Sengupta, and R.M. Welch, "Cloud Field Classification Based upon High Spatial Resolution Textural Features: 2. Simplified Vector Approaches," *J. Geophys. Res.*, 94, No. D12, 14749-1

[6] D. J. Diner, C. J. Bruegge, J. V. Martonchik, G. W. Bothwell, E. D. Danielson,E. L. Floyd, V. G. Ford, L. E. Hovland, K. L. Jones, M. L. White, "A Multiangle Imaging SpectroRadiometer for Terrestrial Remote Sensing from the Earth Observing System," Int. J. of Imaging Systems and Technol., Vol. 3, 92-107, 1991.

[7] J. H. Friedman, J. W. Tukey, "A Projection Pursuit Algorithm for Exploratory Data Analysis," *IEEE Trans. Computers*, Vol. c-23, No. 9, pp. 881 - 890, 1974.

[8] J. H. Friedman, "Exploratory Projection Pursuit," J. of the Amer. Stat. Assoc., March, 1987, Vol. 82, No. 397, Theory and Methods, pp. 249-266.

[9] T. M. Heskes, E. T. P. Slijpen, B. Kappen, "Cooling Schedules for Learning in Neural Networks," Phys. Rev. E, Vol. 47, No. 6, pp. 4457-4464.

[10] P. J. Huber, "Projection Pursuit," *Annals of Statistics*, Vol. 13, No.2, 435-475, 1985.

[11] N. Intrator, L. N. Cooper, "Objective Function Formulation of the BCM Theory of Visual Cortical Plasticity: Statistical Connections, Stability Conditions," *Neural Networks*, Vol. 5, pp. 3-17, 1992.

[12] M. D. Richard, R. P. Lippman, "Neural Network Classifiers Estimate Bayesian a posteriori Probabilities," *Neural Computation*, 3, 461-483, 1991.

[13] G. Vane, R. O. Green, T. G. Chrien, H. T. Enmark, E. G. Hansen,W. M. Porter, *The Airborne Visible/Infrared Imaging Spectrometer (AVIRIS)*, Remote Sens. Environ., 44: 127-143 (1993).

[14] R. M. Welch, S. K. Sengupta, A. K. Goroch, P. Rabindra, N. Rangaraj, M. S. Navar, "Polar Cloud and Surface Classification Using AVHRR Imagery: An Intercomparison of Methods," *J. Appl. Meteorology*, Vol. 31, pp. 405-420, May, 1992.

[15] J. S. Weszka, C. R. Dyer, and A. Rosenfeld, "A comparative Study of Texture Measures for Terrain Classification," *IEEE Trans. Syst., Man, Cybern.*, SMC-6, 269-285, 1976.

OPTIMAL FEATURE EXTRACTION TECHNIQUES TO IMPROVE CLASSIFICATION PERFORMANCE, WITH APPLICATION TO SONAR SIGNALS

Michael J. Larkin
Naval Undersea Warfare Center (Code 3111)
1176 Howell Street
Newport, RI 02841-1708

larkin_m@vsdec.npt.nuwc.navy.mil

Abstract

Feature extraction is an important preliminary step to classification of complex signals. By reducing a high-dimensional signal to a lower-dimensional feature set which preserves the relevant structure of the signal, classification performance is enhanced. A classification system was developed to classify sonar signals as to whether the object detected is minelike or non-minelike. Results are presented comparing classification performance when various feature extraction methods are implemented.

INTRODUCTION

Our objective is to extract pertinent features from acoustic signals in order to develop an automated classifier for sonar signals. The specific application in this case is classification of underwater mines, but this technique could be readily applied to other types of sonar signals. We have used sonar images of the sea bottom which contain sonar returns generated by mines of various types. The purpose of the classifier is to confirm the presence of a mine, and then to ultimately determine what type of mine is present. The emphasis in here is not the classifier itself, but the process of feature extraction which produces a projection of the signal in a lower dimensional feature space, thus improving classifier performance.

The results that will be presented in this paper show successful implementation of feature extraction algorithms in a system for sonar signal classification. Results are described for a two-class (mine and rocks) and four-class (three mine types and rocks), with a comparison of results using several methods of preprocessing.

FEATURE EXTRACTION

In many of the classification systems currently used, the process of feature extraction is relatively ignored in comparison to the method of classification itself. This is due to the fact that in many classification techniques, neural networks in particular, the process of feature extraction is inherently embedded in the classification technique rather that being apparent as a separate process. If a multi-layer neural network is used to classify unprocessed data, the input layer, which learns from examples, will essentially become a feature extractor. Also, the increased availability of computational power enables complex data to be classified with less preprocessing.

However, in problems such as machine vision, speech identification or the problem posed here of detecting objects in sonar returns, the input dimensionality of the problem becomes an impediment to classification. Even neural networks are limited by the problem of parameter estimation- as the number of parameters increases, the size of the data required to train the network increases faster. For a complex problem, obtaining the necessary data may be expensive or even impossible.

Much has been written about this dilemma, known commonly as the *curse of dimensionality*[1]. Because of the inherent sparsity of high-dimensional spaces, an inordinate amount of training data is required to obtain reasonably low variance estimators. The objective of feature extraction is to reduce the dimensionality of the data before performing classification. This is based on the assumption that the important structure in the data actually lies in a much lower dimensional space. If the transformation from the high dimensional space to the low dimensional space is accomplished without losing much relevant information, classification performance will be enhanced.

Thus the optimal method of classification of high dimensional data is a two step process: feature extraction which projects the original data to a lower dimensional feature space; and then performing classification in the low dimensional space. An important consideration is that the features obtained must be independent of class membership, because at this point in the process the class to which the data belongs is not yet known.

DESIGN OF THE FEATURE EXTRACTION/CLASSIFICATION SYSTEM

An overall scheme for preprocessing a given acoustic signal (each sonar return consisting of either a minelike object or a non-minelike object such as a rock), reducing it to a set of features, and using the feature set as an input to an

automated classifier was developed an implemented. Various types of learning algorithms and signal processing techniques were used to perform the preprocessing. These methods include Bienenstock, Cooper and Munro (BCM) learning theory, principal components analysis, and several types of transforms (Karhunen-Loeve, FFT, and wavelets). Results for some of these techniques are described here; other results will be available shortly and will be included in the final paper.

THE BCM ALGORITHM

Modification of synaptic effectiveness between cortical neurons is widely believed to be the physiological basis of learning and memory, and there is evidence that similar synaptic plasticity in many areas of mammalian cortex. In 1982, Bienenstock, Cooper and Munro[2] proposed a concrete synaptic modification hypothesis in which two regions of modification (Hebbian and anti-Hebbian) were stabilized by the addition of a sliding modification threshold. The BCM theory was originally created to explain the development of orientation selectivity and binocular response in various visual environments in kitten striate cortex, one of the most thoroughly studied areas in neuroscience.

In the BCM model, a change in the weight of a synapse is equal to the product of the presynaptic activity and a certain function (ϕ) of the postsynaptic activity. The qualitative consequences of the theory follow from a few properties of the ϕ function:

-when the postsynaptic activity is above the modification threshold (Θ_m), ϕ is positive (i.e. an active synapse will be potentiated);
-when postsynaptic activity is below Θ_m, ϕ is negative (i.e. an active synapse will be depressed);
-there is no activity ($\phi = 0$) when postsynaptic activity is equal either to the spontaneous firing rate or to Θ_m;
-the value of Θ_m is not fixed, but rather "slides" as a function of the postsynaptic activity.

It is the selectivity of BCM neurons that makes this approach so suitable to the problem of feature extraction. In a BCM network, a small number (say, six) of BCM neurons are connected, with lateral inhibition introduced into the connections. In the architecture used here, the kth BCM neuron is inhibited by the 1st through (k-1)th neurons. Thus, a small network of BCM neurons is capable of providing a set of distinct features from an input signal. The usual approach is to train the neurons on a combined set of samples of each type of data (i.e., in an n-

class classification problem, train on all n classes together). The class separability characteristic inherent in BCM neurons yields a set of features which tends to be very effective in identification of the distinct classes.

Previous research presented by the author [3] has demonstrated the capability of the BCM learning rule to perform feature extraction from acoustic signals. In this study, both single BCM neurons and networks of laterally-inhibited BCM neurons were trained on samples of marine mammal sounds. The result of this process is a weight vector (or set of vectors in the multiple neuron case) which, when convolved with a signal of interest, constitutes "testing". The output of the convolution process can then be input into an automated classifier (feed-forward neural network, nearest neighbor, etc.). The results obtained by testing on two types of marine mammals (porpoise and dolphin) demonstrated the capability of BCM neurons to become distinctly selective to one type of signal

MODEL DESCRIPTION

The design of the BCM-based classifier is depicted in Figure 1 (the overall schematic for classifier systems employing other preprocessing techniques is essentially the same). A network of laterally-inhibited BCM neurons is constructed; the number varies depending on the problem, but in the work described here 6 neurons were used. A parameter that adjusts the degree of inhibition between neurons is set; the optimal degree of inhibition is generally determined by performing several training runs and observing how well the value of Θ_m converges as this parameter is adjusted.

Examples of data from each class are used to train the BCM network. As discussed previously, this is an unsupervised procedure- the classes of the data must be assumed unknown at this point. Training the neurons on only one type of data will result in features that are dependent on the class of data used for training- thus they will not be effective to achieve separability between classes. Thus, if we are attempting to ascertain if a sonar return is of a mine or a rock, we must train the BCM network on a combined set of mine and rock data.

The result of this process is that a set of BCM weight vectors is produced. The number of vectors will be the same as the number of BCM neurons in the network; the length of the vector is of a predetermined length, as appropriate for the type of signal. When using acoustic data, the procedure is to sample each set

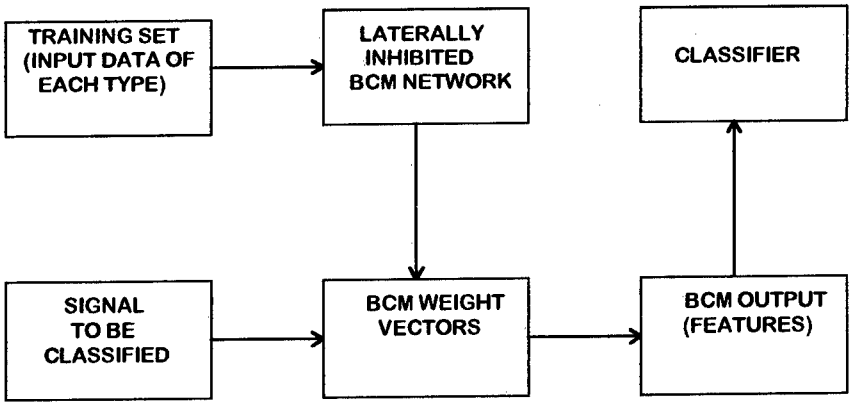


Figure 1. Schematic of BCM-based classifier system

of data randomly by running a window over it (typically of length 512 or 1024) and obtaining a vector of that length.

Next. the dot product of the signal to be tested and the set of BCM weight vectors is computed. The resultant vectors comprise the feature set to be used for classification. In all of the experiments described in this paper, a feed-forward neural network. using a standard backpropagation algorithm. was used. The intention here is not to focus on the classifier itself, but the process of constructing the inputs to the classifier. In order to train the classifier, the testing of the BCM network is first done with data of known classification: then the procedure is done with the trained classifier to determine the classification of an unknown sample.

EXPERIMENTAL RESULTS

The above described procedure was performed with various sets of real data (obtained from surface ship sonars) and sets of synthetic data. In order to provide a benchmark for classifier performance, many of the tests were repeated using wavelet or FFT preprocessors as well as BCM feature extraction: all with classifiers of comparable architecture. The results in the comparison studies using these three techniques were that BCM consistently performed as good as or better than wavelet preprocessing; both methods were consistently better than FFT preprocessing.

Figure 2 shows the results of testing on a set of data obtained by using sets of sonar returns from minelike and non-minelike objects that were denoised as much

as possible; then noise was incrementally introduced to the signals to produce a set of sonar returns of varying SNR. The classification results using BCM, wavelet. and FFT preprocessing were plotted versus the SNR for the various signals. In this example, BCM produced the best results in the mid-range of the curve. While the actual SNR values here might not be the same as a "real" mine classification problem. it is the middle of the curve that is most significant. When the SNR is close to zero, no classifier is going to do much better that a random guess. When the SNR is high, any classifier will have a good chance of performing well. but real sensors are generally not able to achieve this level of SNR, particularly in the shallow water environments of interest to the Navy.

The technique can also be applied to problems with more that two classes. In one experiment. samples of returns were used containing three types of mines (two bottom mines and one close-moored mine) as well as

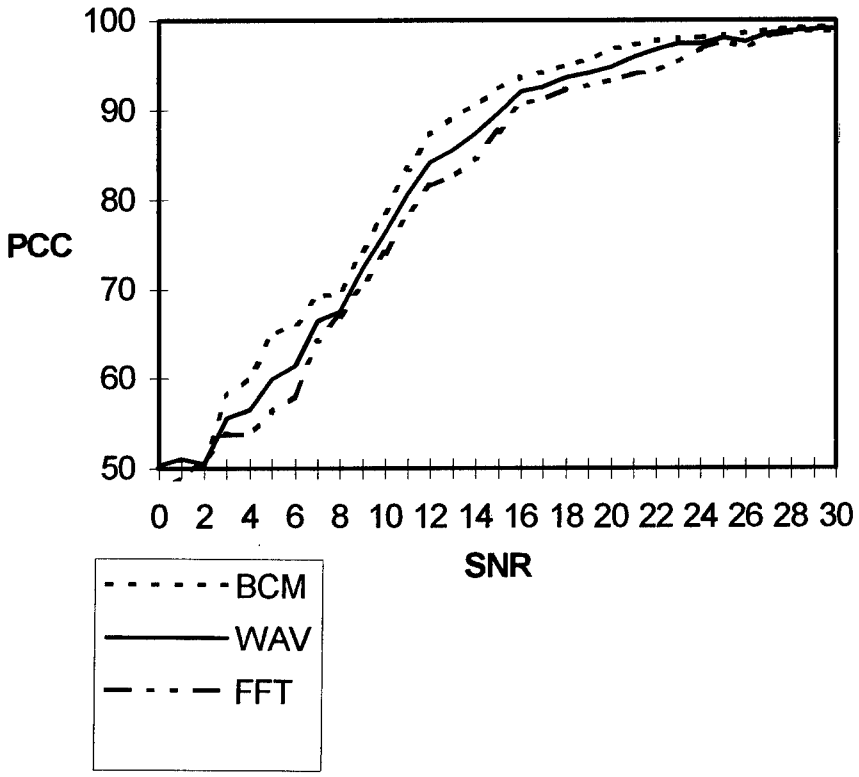


Figure 2. Comparison of Percent Correct Classification of Mines vs. SNR for BCM, wavelet and FFT preprocessing

	B1	B2	CM1	ROCK
B1	0.873	0.087	0.016	0.024
B2	0.085	0.882	0.018	0.015
CM1	0.012	0.036	0.942	0.010
ROCK	0.040	0.032	0.017	0.911
	B1	B2	CM1	ROCK
B1	0.592	0.087	0.154	0.167
B2	0.160	0.622	0.142	0.076
CM1	0.144	0.085	0.688	0.083
ROCK	0.104	0.206	0.016	0.674

Table 1. Comparison of four class problem with BCM (top) and FFT (bottom) preprocessing

rocks. The SNR of the signals is not considered in this problem. The confusion matrix of the classification results, comparing BCM with FFT preprocessing, is shown in Table 1. The percent of correct classification when BCM was employed ranged from 87% to 94% among the classes; when FFTs were used the results ranged from 59% to 69% correct.

CONCLUSIONS AND ONGOING RESEARCH

We have developed and demonstrated an automated classifier for acoustic signals using the BCM learning procedure to perform feature extraction. The results obtained in the various classification problems support the contention that the BCM algorithm is an effective means of obtaining features for the purposes of classification. We will continue to run similar classification problems using the various feature extraction methods to determine the optimal method, which might depend on the type of signal to be classified.

Ongoing work is emphasizing the multiple class problem, and comparing the approach of using a one or two levels of classification. The multiple mine type classifier may be posed as a two level classification problem, as a sonar return from a submerged object is first classified as minelike or non-minelike, and then minelike objects are identified as a particular type of mine.

A particular area of interest at present is determining the characteristics of the features that the BCM neurons identify in minelike objects. For example, a trained sonar operator determines that an object is minelike by observing several characteristics (size, shape, reverberation, shadow, aspect) in the sonar image. We are presently attempting to ascertain if the features derived by BCM are analogous.

Another method that promises to achieve good classification performance is using genetic algorithms to obtain the best feature set for the classifier. Our work with this approach is too preliminary to present results at the moment. and will be presented in a follow-on paper.

REFERENCES

[1] R.E. Bellman. *Adaptive Control Processes*. Princeton University Press, 1961.

[2] E.L. Bienenstock, L.N. Cooper, & P.W. Munro. Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex, *Journal of Neuroscience*, 2:32-48, 1982.

[3] M.J. Larkin. Feature extraction in acoustic signals using the BCM learning rule, *Proceedings of the IEEE Asilomar Conference on Signals, Systems & Computers*. Pacific Grove. CA. pp. 889-893. November 1995.

ON ESTIMATION OF NONLINEAR BLACK-BOX MODELS: HOW TO OBTAIN A GOOD INITIALIZATION

Jonas Sjöberg
Department of Applied Electronics, Chalmers University of Technology,
412 96 Gothenbourg, Sweden, Fax: +46-31-7721782,
Email: sjoberg@ae.chalmers.se

Abstract. An algorithm how to define and initialize nonlinear recurrent models using linear models is described. From a modeling point of view it is natural to try linear models first and then continue with nonlinear models. The suggested method gives such an algorithm and the nonlinear recurrent model is defined as an extension of the linear model. This gives less problems with local minima compared to a random initialization. Also, the stability of the model and its derivative with respect to the parameters can be guaranteed which is a requirement for the prediction-error estimation method (sometimes called back-propagation through time) to be applicable.

1. INTRODUCTION

Parameter estimation of recurrent neural networks is often described as a tricky subject. See, *e.g.*, [3]. The problem can be described as an iterative criterion minimization. Depending on the initial parameter guess the solution converges to different local minima. With a good initial parameter guess the chances increase to converge to the global minimum, or at least to a favorable local minimum. Recurrent models are especially sensitive to the initial parameter values and convergence to a good minimum cannot be expected unless two filters are stable. The first filter is the predictor model and the second one is the derivative of the prediction with respect to the model parameters.

In this contribution we have a slightly different approach than in many other works on recurrent networks. We do not have a specific recurrent network in focus which is investigated and applied to different problems. Instead, the approach is more problem oriented in an engineering point of view. Given input-output data from a dynamic system (or just output data in case of a time series) the goal is to obtain a model which describes the system as good as possible and in this search all kind of different models can be considered.

0-7803-4256-9/97/$10.00

In the search for a good model structure (*i.e.* network) one might want to try some recurrent models. Then the topic of this work becomes relevant: initialization of nonlinear recurrent models (*i.e.* recurrent networks).

Since the parameter estimate has to be computed by an iterative search for the minimum of a criterion of fit there will usually be problems with local minima. Already with linear recurrent models one has such problems and a good initial parameter guess is important to assure convergence of the iterative search to the global minimum. For nonlinear models the problems with local minima will be even more serious.

Example 1 *In Figure 1 a) the true step response of a linear second order system is shown together with the response of a model corresponding to a local minimum. The true system is oscillating and its complex poles are close to the unit circle, Figure 1 b). The model has a real pole in between the true poles and its second pole is just outside the unit circle. This local minimum corresponds to a model which has only modelled the mean value of the output signal.*

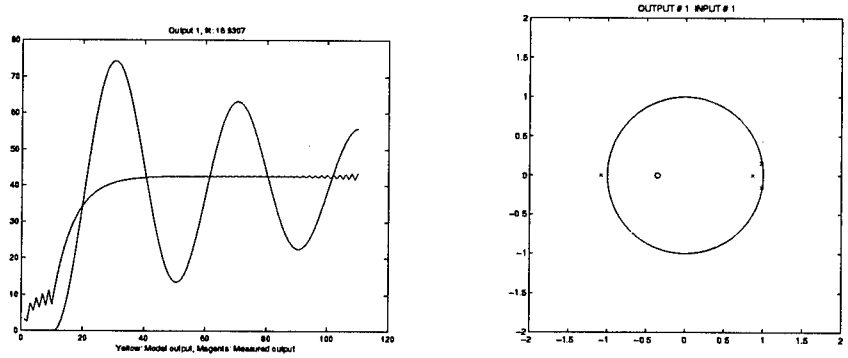


Figure 1: A) Step response of a second order linear system and a model corresponding to a local minimum. b) The two complex poles (x) of the linear system at the unit circle and those of the model on the real axis.

By starting with a linear model and adding a nonlinearity to it, the linear model gives a clue how to choose the parameters in the nonlinear part. It has to be decided upon the position of the basis functions and they should be placed so that they are activated by the estimation data. Otherwise the nonlinearity will not have any influence on the model. To make this decision one needs a "preliminary" model. Since there exist well developed algorithms and tools for linear models and since linear models perform well, or fairly well, on many problems it is natural to choose the preliminary model to be linear. There is, however, no problem to apply the algorithm to an existing nonlinear model and then obtain a more advanced nonlinear model. In addition to the position of the basis function one also has to decide upon the amplitude. By choosing it to zero the initial nonlinear model becomes equal to the linear model. The advantage of this is that stability of the two filters follows from the

stability of the linear model. Also, the parameters are calculated by a gradient based minimization starting at a point where the performance equals that of the linear model. This means that the performance of the nonlinear model cannot become worse than that of the linear model. Hence, the algorithm guarantees a nonlinear model which performs better (on estimation data) than the linear model.

The algorithm is quite general and can be used to obtain nonlinear extensions to any linear model class. Recurrent models are equivalent to state-space models, see, *e.g.*, [7], so the algorithm presented here can be seen as a procedure how to obtain nonlinear black-box state-space models. A subset of the recurrent models can also be described as Output Error (OE) and ARMAX models. Hence, the algorithm can also be used to construct and estimate nonlinear OE and ARMAX models.

To the authors best knowledge there is very little work done addressing the initialization of recurrent models. Typically, the parameters are initialized with small random values and the suggested method will be compared to the random choice on a small example in Section4.

The paper is organized as follows. A short background and a problem definition is given in Section 2. Then the initialization algorithm and the stability of the two nonlinear filters are given in Section 3. The suggested method is applied to a small example in Section 4 and the paper is concluded in Section 5.

2. PROBLEM DESCRIPTION

Let $y(t)$ be the output of the process to be modeled and $u(t)$ the input signal which influences the process. For simplicity it will be assumed that both $y(t)$ and $u(t)$ are scalars. It is straight forward to extend the results to the multi-input and multi-output case. For time series where no input is used the results follows right away by just canceling $u(t)$ in all equations.

The goal is to find a model which uses past measurements to predict future outputs $y(t)$. A common black-box approach is to consider a parameterized candidate model

$$\hat{y}(\theta, t) = g(\theta, \varphi(t, \theta)) \tag{1}$$

where $\hat{y}(t)$ is the prediction of $y(t)$, θ is the parameter vector to be tuned, and $\varphi(t, \theta)$ is the regressor which contains information available at time t. In this way the modeling has been divided into two parts, a choice of regressor φ and a choice of mapping g. Depending on the choice of $\varphi(t, \theta)$ and g different models, or neural nets, are obtained. A short background on this is given here, see [9] for a deeper discussion.

2.1. The regressor

The regressor $\varphi(t, \theta)$ is formed by a mapping from data

$$\{y^{t-1},\ u^{t-1}\} \longrightarrow \varphi(t, \theta) \in \mathbf{R}^d,$$

where $y^{t-1} = [y(1), \ldots, y(t-1)]$ and u^{t-1} is defined analogous. If φ depends on θ the model is *recurrent*, *i.e.*, the regressor contains information given by the model at earlier time instants. This can be described in several different ways, *e.g.*, by using states

$$\begin{bmatrix} x(t+1) \\ \widehat{y}(t) \end{bmatrix} = g(\theta, \varphi(t,\theta)) \tag{2}$$

where

$$\varphi(t,\theta) = \begin{bmatrix} x(t) \\ u(t) \end{bmatrix}$$

and $x(t)$ are the *states* of the model, *i.e.*, all information which has to be stored at each time step and then fed into the model at the next time instant. The function g is now a function $\mathbb{R}^{d+1} \to \mathbb{R}^{d+1}$ where d is the number of states. The description (2) includes all recurrent models like, *e.g.*, Elman and Jordan networks, see [7] for further explanation.

Also input-output models can be described like (1) with components of the regressor φ chosen in analogy with their linear counterparts. See [9].

- FIR $\varphi(t)$ consisting of $u(t-k_u)$,
- ARX $\varphi(t)$ consisting of $y(t-k_y), u(t-k_u)$,
- OE $\varphi(t,\theta)$ consisting of $\widehat{y}(t-k_y), u(t-k_u)$
- ARMAX $\varphi(t,\theta)$ consisting of $y(t-k_y), u(t-k_u), \varepsilon(\theta, t-k_y)$

where $\varepsilon(\theta,t) = y(t) - \widehat{y}(t,\theta)$, $k_y = 1, 2\ldots$ and $k_u = 0, 1\ldots$.

The models where $\varphi(t,\theta)$ depends on $x(t)$ or $\widehat{y}(t,\theta)$, *i.e.*, the state-space models, OE, and ARMAX models, are called *recurrent*.

Linear models are obtained by choosing g to be a linear mapping. Note that if the model is recurrent or not depends only on the choice of regressor, φ, and not on the mapping g. Hence, the recurrent regressors mentioned above also give recurrent models when g is chosen to be linear.

In the following the input-output form (1) of the model will be used in the discussions. It is straight-forward to modify the method to state-space models.

2.2. The Mapping

The mapping $g(\cdot,\cdot)$ in (1) can be any parameterized function. Most black-box models can be described as a basis function expansion

$$\widehat{y}(t) = g(\theta, \varphi(t,\theta)) = \sum_{i=1}^{n} c_i g_i(\varphi(t,\theta), a_i) \tag{3}$$

$$\theta = [c_1 \; a_1 \; c_2 \; a_2 \; \ldots c_n \; a_n].$$

Depending on the specific choice of the functions $g_i(\varphi, a_i)$ we obtain different model structures like neural nets, radial basis functions etc. The basis function expansion can also be mixed with different types of functions $g_i(\varphi, a_i)$ and here we will consider the case when the first function in the expansion is linear. Then(3) can be expressed as

$$\widehat{y}(t) = g(\theta, \varphi(t,\theta)) = \theta_l^T \varphi_l(t,\theta) + \sum_{i=1}^{n} c_i g_i(\varphi_{nl}(t,\theta), a_i) \tag{4}$$

where $\theta_l^T \varphi_l(t,\theta)$ expresses the linear model which will be is used to initialize the parameters in the second, nonlinear term. Subscript l denotes linear- and nl nonlinear part. This gives us the freedom to use different regressors in the different parts.

2.3. Calculate the Parameter Estimate

Given a model structure g and an estimation data set $\{y(t), u(t)\}_{t=1}^{N}$ the parameter estimate $\widehat{\theta}_N$ is defined as the minimum of a criterion of fit, *e.g.*, sum of squared errors

$$\widehat{\theta}_N = \arg\min_{\theta} V_N(\theta)$$

where

$$V_N(\theta) = \frac{1}{N} \sum_{t} \varepsilon^2(\theta, t) \tag{5}$$

and

$$\varepsilon(\theta, t) = y(t) - \widehat{y}(\theta, t). \tag{6}$$

The criterion (5) is typically minimized by an iterative search based on the gradient of the criterion. See, *e.g.*, [1, 10]. From an initial parameter value θ_0 the criterion (5) is stepwise decreased by changing the parameter vector in a descent direction of the criterion until a minimum is reached.

The derivative of the criterion (5) becomes

$$\frac{dV_N(\theta)}{d\theta} = -\frac{2}{N} \sum_{t=1}^{N} \varepsilon(\theta, t) \frac{dg(\theta, \varphi(t,\theta))}{d\theta} \tag{7}$$

where the derivative of the model output (1) is

$$\frac{dg(\theta, \varphi(t,\theta))}{d\theta} = \frac{\partial g(\theta, \varphi(t,\theta))}{\partial \theta} + \sum_{i=1}^{\dim \varphi} \frac{\partial g(\theta, \varphi(t,\theta))}{\partial \varphi_i(t,\theta)} \frac{d\varphi_i(t,\theta)}{d\theta}. \tag{8}$$

The second term is non-zero only in the case the model is recurrent. Then $\varphi(t,\theta)$ contains components of $g(\theta, \varphi(\tilde{t}))$, $\tilde{t} < t$. This makes (8) a nonlinear filtering with input signal

$$\frac{\partial g(\theta, \varphi(t,\theta))}{\partial \theta}.$$

Before the filtering (8) can be performed $\varphi(t,\theta)$ must be obtained. Hence also the filter

$$\hat{y}(t,\theta) = g(\theta, \varphi(t,\theta)) \tag{9}$$

with input $u(t)$ must be performed.

The success of the iterative minimization of $V_N(\theta)$ (5) (or learning) depends on the initial parameter guess θ_0. A better θ_0 means that the parameter estimate converges to a better local minimum of $V_N(\theta)$. Moreover, as described in [4] the nonlinear filters (8) and (9) must be stable for a succesful minimization.

3. INITIALIZING THE NONLINEAR MODEL

Here follows a presentation of a "natural" first step from a linear model to a nonlinear model. Consider a basis function expansion like (4) and let the linear model be the first basis function. If the model is non-recurrent the parameters c_i can be fitted by least squares and the nonlinear model will be better than the linear model already at the initialization. For recurrent models the parameters c_i are initially set to zero. This gives an initialization of the nonlinear model which is identical to the linear model. It will be shown that the stability of the filters (8) and (9) then follows from the stability of the linear model. Of special importance is the fact that the linear model can be used to obtain a $\varphi_{nl}(t,\theta)$ which is necessary to obtain a good initialization of the parameters a_i in (4).

Proceed the modelling as follows:

- Start with a linear model as depicted in Figure 2 a) (but with, possibly, different regressor) and estimate its parameters.
- Complement the linear model with nonlinear elements, for example as shown in Figure 2 b). Initialize the nonlinear part and calculate $\hat{\theta}_N$ by minimizing $V_N(\theta)$.

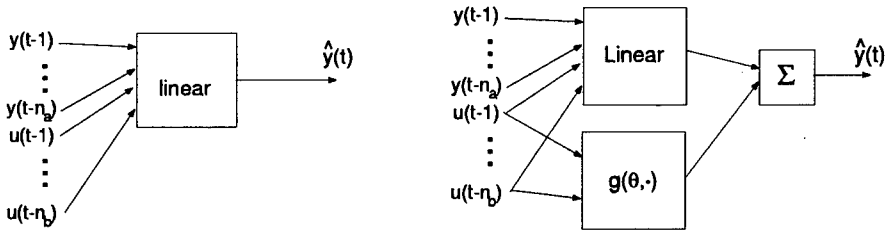


Figure 2: a) Linear model. b) Linear model complemented with a nonlinearity.

This approach gives us a model structure like the one in (4). In case a state-space model (2) is used the added nonlinearity can be chosen to influence the model output $\hat{y}(t)$ or one of the states $x(t)$.

The nonlinear element can have a regressor φ_{nl} which differ from the linear regressors φ_l. In this way the nonlinear model can be chosen to be linear with

respect to some regressors. Such models will be less nonlinear but have other nice features. See [6, 9].

When the structure is decided upon it is time to initialize the nonlinear model. There are three kinds of parameters in the model (4) which have to be initialized:

- θ_l: Linear part of the model.
- a_i: Parameters defining localization of basis function. The model is nonlinear in these parameters.
- c_i: Amplitude of basis functions. Model linear in these parameters.

The following algorithm specifies how the parameters are initialized.

Algorithm 1 *The parameters are initialized as:*

- θ_l: *given by the linear model.*
- a_i: *chosen randomly but with a probability distribution so that basis functions placed on the support of regressor $\varphi_{nl}(t,\theta)$. The linear model is used to construct $\varphi_{nl}(t,\theta)$. See [8].*
- c_i: *for non-recurrent models they are calculated by least squares on the residuals $\varepsilon = y - \theta_l^T\varphi$ with $g_i(\varphi_{nl}(t,\theta),a_i)$ as regressors[1]. If the model is recurrent then c_i are set to zero.*

The following two theorems reveal the advantage of this initialization.

Theorem 1 *The nonlinear non-recurrent model (4) initialized as described in Algorithm 1 gives better fit than the linear model.*

Proof. Let $\varepsilon(\theta,t) = y(t) - \theta_l^T\varphi(t,\theta)$ be the residuals of the linear model. Sum of squared errors of the initialized nonlinear model then becomes

$$\sum_{t=1}^{N}(\varepsilon(\theta,t) - \sum_{i=1}^{n} c_i g_i(\varphi_2(t,\theta),a_i))^2$$

since c_i are chosen to minimize this sum the following inequality holds

$$\sum_{t=1}^{N}(\varepsilon(\theta,t) - \sum_{i=1}^{n} c_i g_i(\varphi_2(t,\theta),a_i))^2 \leq \sum_{t=1}^{N}\varepsilon^2(\theta,t)$$

with equality if all $c_i = 0$. □

Theorem 2 *The nonlinear recurrent model (4) initialized as described in Algorithm 1 gives a fit equally good as the linear model. Moreover, the nonlinear filters (8) and (9) are stable if the linear model is stable.*

[1] It is also possible to fit θ_l together with c_i

Proof. Notice that $c_i = 0 \Rightarrow g(\theta, \varphi) = \theta_i^T \varphi$ for all φ, *i.e.*, the nonlinear model is reduced to the linear model. From this the stability of the filter (9) follows. The stability of filter (8) depends on

$$\frac{\partial g}{\partial \varphi}$$

and using $c_i = 0$ one obtains from (4)

$$\frac{\partial g(\theta, \varphi(\theta,t))}{\partial \varphi(\theta,t)} = \frac{\partial \theta_i^T \varphi_l(\theta,t)}{\partial \varphi(\theta,t)} = \theta_l.$$

Since this is identical to the linear model the stability follows. □

Hence, if the linear model is stable then the initialized nonlinear filters also become stable. From this initialization the criterion $V_N(\theta)$ is iteratively minimized as described in Section 2.3.

The suggested method can be modified and extended in a number of different ways, for example

- The same procedure can be used to add new nonlinearities to an existing nonlinear model.
- The idea can be used to initialize Elman networks, [2], and other models which are close to linear if the data are properly scaled. The nonlinear model can then be initialized approximately equal to the linear model.

4. EXAMPLE

The suggested algorithm will now be applied to a small problem to illustrate the advantages compared to random initialization of the model parameters.

The data are generated in the following way. The input signal $u(t)$ consists of 110 samples with unit step at sample 10. Then the output $y(t)$ is obtained by filtering the input signal through the filter

$$y(t) = f(y(t-1), y(t-2), u(t-1)) \tag{10}$$

where $f(\cdot)$ is one-hidden-layer sigmoidal neural net with two sigmoids. The parameters are chosen so that $y(t)$ becomes oscillating but cannot be models satisfactory by a linear model. The output signal is shown in Figure 3 b).

Two different approaches to model the data are tested. First a neural net model of the same type as that one which generated the data is tried

$$\hat{y}(t,\theta) = g(\theta, \hat{y}(t-1), \hat{y}(t-2), u(t-1)) \tag{11}$$

where g is the network and θ its parameters. Several different initial parameters chosen randomly from Gaussian distributions with different variances are tried. From the different initial values the criterion of fit (5) is minimized with the Levenberg-Marquardt algorithm. The equation (8) is used to obtain the derivative of the model. For on one of the initial parameters values

the criterion decrease as a function of the number of iterations is shown in Figure 3 a). The simulation of the obtained model is depicted in Figure 3 b) together with the true output. Obviously the network has not been able to model anything else than just the mean level of the signal. Although a larger number of different initial parameter values were tried none of them performed any better than the depicted result.

The second approach starts with a linear model

$$\hat{y}(t,\theta) = f_1\hat{y}(t-1) + f_2\hat{y}(t-2) + b_1u(t-1).$$

To estimate the parameters f_1, f_2, b_1 of the linear model one can use standard methods which are good to avoid local minima. See, *e.g.*, [4, 5]. The simulation with this linear model is also depicted in Figure 3 b). Although it is not able to described the output very well, it describes the oscillations fairly well. The estimated linear model is then complemented with a nonlinear part which equals the model (11) and the parameters are estimated according to Algorithm 1. Since there is some randomness in the position of the basis functions a_i the procedure is repeated several times giving a set of different initial parameter values. The same numerical minimization algorithm as for the randomly initialized network is then used to calculate the minimum of the criterion. A typical result is shown in Figure 3. In a) the criterion decrease is shown and in b) the simulation. The performance is close to perfect and it would be come even better with more iterations of the search algorithm. Non of the models obtained with this initialization was stuck in any local minimum.

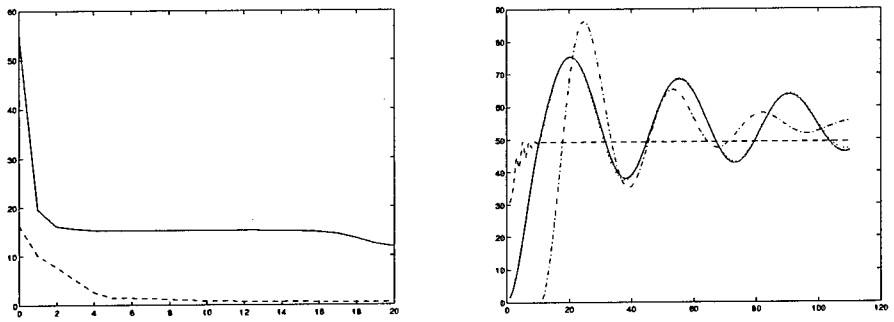


Figure 3: A) Criterion decrease as a function of the number of iterations. Solid line: random initialization. Dashed line: The suggested method. b) Solid line: True step response. Dashed: random initialized model. Dashed-dotted: linear model used to initialize the nonlinear model. Dotted: nonlinear model obtained with the suggested initialization (hardly visible since it is very similar to the solid line.)

5. CONCLUSIONS

It has been shown that

- There is a "natural" way to define and initialize nonlinear recurrent models so that they perform as good as a linear model already at the initialization. The performance is then improved further by a numerical minimization of the criterion of fit.
- Problems with local minima are likely to be smaller with the suggested method.
- The suggested scheme guarantees stability of the two nonlinear filters which are necessary to minimize the criterion of fit by a gradient search.

REFERENCES

[1] J.E. Dennis and R.B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Prentice-Hall, Englewood Cliffs, New Jersey, 1983.

[2] J.L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.

[3] S. Haykin. *Neural Networks: A Comprehensive Foundation.* Macmillan College Publishing Company, 866 Third Avenue, NY, 1994.

[4] L. Ljung. *System Identification: Theory for the User.* Prentice-Hall, Englewood Cliffs, NJ, 1987.

[5] L. Ljung and T. Glad. *Modeling of Dynamic Systems.* Prentice Hall, Englewood Cliffs, N.J., 1994.

[6] K.S. Narendra and K. Parthasarathy. Identification and control of dynamical systems using neural networks. *IEEE Trans. Neural Networks*, 1:4–27, 1990.

[7] O. Nerrand, P. Roussel-Ragot, L. Personnaz, and G. Drefys. Neural networks and nonlinear adaptive filtering: Unifying concepts and new algorithms. *Neural Computation*, 5:165–199, 1993.

[8] J. Sjöberg. Nonlinear black-box structures – some approaches and some examples. In M. Jamshidi and P. Dauchez, editors, *Proc. Second World Automation Congress, Montpellier, France, May*, volume 4, pages 679–684, 1996.

[9] J. Sjöberg, Q. Zhang, L. Ljung, A. Benveniste, B. Deylon, P-Y. Glorennec, H. Hjalmarsson, and A. Juditsky. Non-linear black-box modeling in system identification: a unified overview. *Automatica*, 31(12):1691–1724, December 1995.

[10] P.P. van der Smagt. Minimisation methods for training feedforward neural networks. *Neural Networks*, 7(1):1–11, 1994.

INTERPRETATION OF RECURRENT NEURAL NETWORKS

Morten With Pedersen and Jan Larsen
CONNECT, Department of Mathematical Modelling, Building 321
Technical University of Denmark, DK-2800 Lyngby, Denmark
Phones: + 45 4525 + ext. 3920,3923 Fax: + 45 45872599
emails: mwp@imm.dtu.dk, jl@imm.dtu.dk

Abstract - This paper addresses techniques for interpretation and characterization of trained recurrent nets for time series problems. In particular, we focus on assessment of effective memory and suggest an operational definition of memory. Further we discuss the evaluation of learning curves. Various numerical experiments on time series prediction problems are used to illustrate the potential of the suggested methods.

INTRODUCTION

It is widely recognized that recurrent neural networks (RNNs) are flexible tools for time series processing, system identification and control problems, see e.g., [3]. Feed-forward networks can accommodate dynamics by having a lag space of past input and target values; however, a fully recurrent network with internal feedbacks allows for even more sophisticated dynamics. While fully RNN architectures are the ultimate tool for modeling dynamic relations, the comprehension of the networks is a challenging subject of ongoing research. Theoretical investigations of modeling capabilities of RNNs have been reported, see e.g., [2], [4], [7]. However, to the authors knowledge, there is no general theory of the dynamic behavior of a general RNN except for very special models like the Hopfield network, see e.g., [3]. This indeed indicates that theoretical analysis of RNNs is extremely complicated. On the other hand, one might pursue a more computational approach. The general computational tools from non-linear dynamic systems analysis like phase portraits, stability analysis, measurement of fractal dimensions or Lyapunov exponents (see e.g., [1], [3]) may be applied to the analysis of RNNs.

The motivation for this paper is evaluation and interpretation of *trained* recurrent networks, and to suggest and discuss simple operational techniques. In particular, we focus on the *learning curve* and present a new method to determine the effective *memory* of a recurrent network which conveys the relevant time scale of the dynamics.

0-7803-4256-9/97/$10.00

NETWORK ARCHITECTURE

The objective is to model a non-linear dynamic relation among a discrete-time input signal $x(t)$ and a discrete time target signal, $d(t)$. The general architecture of the RNN considered in this presentation is based on [5] and consists of a single hidden layer of fully connected nonlinear units and one output unit. In particular, we focus on a network with only one external input, viz. the most recent value, $x(t)$. That is, the only information available about previous inputs stems from the memory build up internally in the net. The advantage using these networks is that the tedious problem of determining the optimal lag space of previous inputs is converted into determining the optimal network architecture in terms of connections and number of hidden neurons.

The network has a linear output in order to allow for arbitrary dynamic range, and at time t the prediction of the target $d(t)$ is given by,

$$y(t) = \sum_{i=1}^{N_h} w_{oi} \cdot s_i(t) + w_{ob} \tag{1}$$

where N_h is the number of hidden units, w_{oi} is the weight to the output unit from hidden unit i and w_{ob} is the output bias weight. The ith *state*, $s_i(t)$, is the output of a hidden unit computed as

$$s_i(t) = f\left(\sum_{j=1}^{N_h} w_{ij} \cdot s_j(t-1) + w_{ix} \cdot x(t) + w_{ib}\right) \tag{2}$$

where w_{ij} is the weight to hidden unit i from hidden unit j, w_{ix} is the weight from the external input $x(t)$, and w_{ib} is the bias weight. $f(\cdot)$ is the nonlinear activation function $\tanh(x)$. Note that the update of the units is *layered* [5]: at each time step the hidden units are updated before the output unit.

TRAINING AND GENERALIZATION

Suppose we have a training set of related values of inputs and targets $\mathcal{T} = \{x(t), d(t)\}_{t=1}^{T}$ where T is the number of training samples. Training is done by adjusting the weights so as to minimize a cost function. Here we employ the sum of squared errors augmented by a simple weight decay regularization term

$$C(\mathbf{w}) = \frac{1}{2}\sum_{t=1}^{T} e^2(t) + \frac{\alpha}{2}|\mathbf{w}|^2 \quad , \quad e(t) = d(t) - y(t) \tag{3}$$

where $\mathbf{w}$ is the concatenated set of weights and α is a small regularization parameter. Training aims at minimizing the cost function $C(\mathbf{w})$ and is thoroughly treated for RNNs in [6].

Suppose that training provides the estimated weight vector $\widehat{\mathbf{w}}$. Let π be an initial state vector[1] of the "true" data generating system leading to the training set $\mathcal{T}$ and define an associated probability distribution[2] $p(\pi)$. Further, define $\mathbf{x}(t) = [x(t), x(t-1), \cdots x(T+1)]^\top$ and let $p\left(d(t), \mathbf{x}(t) \mid \mathcal{T}, \pi\right)$, $t > T$, be the true joint probability density function of $[d(t), \mathbf{x}(t)]$ conditioned on the initial state π and the training set $\mathcal{T}$. The true joint p.d.f. is assumed to be time-independent (i.e., stationary). The generalization error of the trained net is defined as the expected squared prediction error on future data *immediately* succeeding the training data, i.e., for $t > T$,

$$G(\widehat{\mathbf{w}}) = \int [d(t) - y(t; \widehat{\mathbf{w}})]^2 \cdot p\left(d(t), \mathbf{x}(t) \mid \mathcal{T}, \pi\right) \cdot p(\pi)\, dd(t) d\mathbf{x}(t) d\pi \qquad (4)$$

Thus the generalization error is the ensemble average of the squared error over 1) possible realizations of $[d(t), \mathbf{x}(t)]$ due to inherent stochastic processes in the data generating system, and 2) over possible initial states leading to the particular training set.

We estimate the generalization error by,

$$\widehat{G}(\widehat{\mathbf{w}}) = \frac{1}{V} \sum_{t=T+1}^{T+V} e^2(t; \widehat{\mathbf{w}}) \qquad (5)$$

where V is the number of test samples.

LEARNING CURVE

The learning curve expresses the average generalization error over all possible training sets of a particular size T as a function of T and is an important tool for verifying whether enough data is available for proper training of the network. Moreover, the shape of the curve provides insight into the nature of the problem as demonstrated in the experimental section.

Practical considerations may lead to more restricted definitions. Here we compute the learning curve as the estimated generalization error when gradually expanding the training set. That is, there is no average over different sets of a particular size.

NETWORK MEMORY

A characteristic of recurrent neural networks is their ability to build up an internal memory representing the "history" of previous inputs on which the predictions of future values is based. The significance of this internal memory is especially clear when using RNNs having only one external input. Without the ability to create internal memory this class of networks would be useless.

Once a recurrent network is trained, the basic idea here is to define an integer variable M which expresses the effective memory of past values of

[1] The initial state captures the all information about the time series for $t \leq 0$.

[2] E.g., that all initial states are equally likely.

the input signal $x(t)$. The memory thus provides a partial insight into the functionality and dynamics of the network. The experimental section gives examples of interpreting the dynamics using this simple concept. Recurrent networks with only one external input can not give individual contribution to each previous input $x(t-m)$ but must store their own representation. Consequently, the RNN has a certain *memory profile*. We are currently pursuing the idea of determining the memory profile.

A feed-forward network does not possess any internal memory, i.e., the memory is explicitly determined by the memory contained in the preprocessing of the input signal. The standard approach is to feed the signals from a tapped delay line $[x(t), x(t-1), \cdots x(t-M)]$ into the network and the memory thus equals M.

The capacity of the internal memory of a recurrent network increases when the number of hidden units (i.e., the dimension of the state vector) increases as the state vector contains all information about previous inputs. However, to our knowledge, there is no reports on quantizing the notion of memory in recurrent networks. In the following we attempt to provide a definition of the memory of a specific trained recurrent network.

The output from the RNN defined in (1), (2) is based on the current and – in principle – infinitely many previous inputs[3], as shown by,

$$y(t) = y\left(t|\widehat{\mathbf{w}}, x(t), x(t-1), \ldots, x(-\infty)\right). \tag{6}$$

In order to determine the effective *average* memory of the recurrent network we suggest to evaluate an estimate of the generalization error, i.e., prediction error on a test set, using predictions based on only a *limited* number of previous inputs. This generalization error is then compared to the error obtained using all – in principle infinitely many – previous inputs.

In particular, when evaluating the generalization error using only the m most recent inputs, we compute,

$$\widehat{G}_m(\widehat{\mathbf{w}}) = \frac{1}{V}\sum_{t=T+1}^{T+V}\left[d(t) - y\left(t|\widehat{\mathbf{w}}, x(t), x(t-1), \ldots, x(t-m)\right)\right]^2,\ m \geq 0 \tag{7}$$

where V is the size of the test set. $y\left(t|\widehat{\mathbf{w}}, x(t), x(t-1), \ldots, x(t-m)\right)$ is computed for each $t \in [T+1; T+V]$ by *resetting*[4] the states $s_i(t-m-1)$, $i = 1, 2, \cdots, N_h$, to zero and then iterate the network from time $t-m$ until time t, using the output $y(t)$ at this time as the prediction of $d(t)$. In the first iteration, calculating $y\left(t-m|\widehat{\mathbf{w}}, x(t-m)\right)$, the network thus functions as a feed-forward network since the previous states of the hidden units – and thereby all previous external inputs – have no influence on the network output. Then, the network gradually builds up a representation of the past in

[3]This is also true for a RNN in which previous values of the output is fed back to the input.

[4]Setting the hidden unit states $s_i(t-m-1)$ to zero is equivalent to erasing the memory of the network regarding inputs before time $t-m$.

the hidden units during the next $m+1$ iterations before it makes its prediction at time t.

The resulting errors $\widehat{G}_0(\widehat{\mathbf{w}}), \widehat{G}_1(\widehat{\mathbf{w}}), \ldots$ are then compared to $\widehat{G}_\infty(\widehat{\mathbf{w}})$ denoting the error obtained when using all available previous inputs, i.e., no resetting of the hidden unit states at any time. The memory M is now defined as,

$$M = \inf\left\{ m \;\middle|\; \forall m' \geq m,\ \frac{|\widehat{G}_{m'}(\widehat{\mathbf{w}}) - \widehat{G}_\infty(\widehat{\mathbf{w}})|}{\widehat{G}_\infty(\widehat{\mathbf{w}})} < \epsilon \right\} \tag{8}$$

where ϵ is a *small* number. Thus, the memory, M, denotes the minimal number of previous inputs beyond which additional inputs are insignificant.

The memory measure outlined above determines the number of previous inputs that the network needs knowledge about in order to obtain good predictions on *all* samples in the test set. Thus the measure can be interpreted as the *average* memory of the network. A recurrent network, however, is a *dynamic* system whose internal characteristics can be highly influenced by the nature of the input series. Especially, if the input series exhibits regions of non-stationary behavior, the network dynamics including memory must clearly be affected. Such changes in dynamics are not captured by the average memory measure and we may define a *local memory*, in accordance with (8), using a local generalization error estimate[5]

$$\widehat{G}_m(t;\widehat{\mathbf{w}}) = \frac{1}{K}\sum_{t'=t-K+1}^{t} [d(t') - y\,(t'|\widehat{\mathbf{w}}, x(t'), x(t'-1), \ldots, x(t'-m))]^2, \tag{9}$$

where $m \geq 0$, $t > T$, and $1 \leq K \leq V$ is the size of a smaller test set. Choosing K too small gives rise to a very noisy measure of the generalization error; however, in principle a good resolution of changes in memory requirement. On the other hand, increasing K improves generalization accuracy but reduces the resolution of changes in memory.

EXPERIMENTS

The proposed methods for estimating the learning curves and memory are evaluated on two chaotic time series prediction problems, viz. the laser series from the Santa Fe time series competition [9] and the artificially generated Mackey-Glass series [8].

The laser series is illustrated in the left panel of Figure 1. Let $z(t)$ denotes the series, then identification is done by training the network to perform a one step ahead prediction, i.e., we use $x(t) = z(t)$ and $d(t) = z(t+1)$. All available 10093 samples are used and scaled to zero mean and unit variance. From these data we construct a learning curve. The training series are obtained by

[5]Notice, by defining this measure for all $t > T$ some of the first values are based partly on training examples.

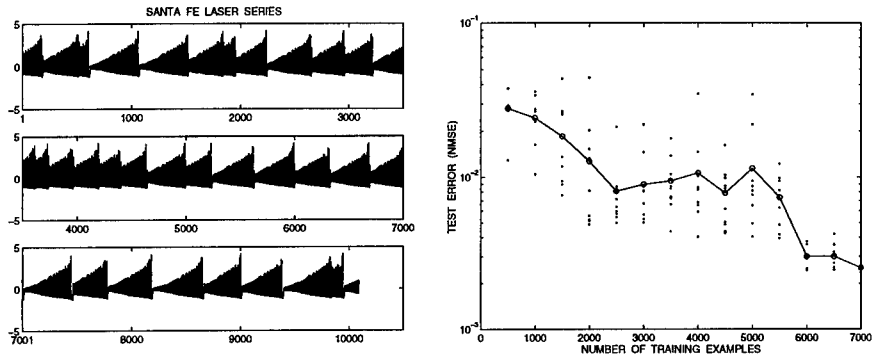


Figure 1: Left panel: The Santa Fe laser series. Right panel: Learning curve for the laser data. Dots denote error for individual nets, the connected circles indicate the average.

extending backwards in time from point 7000 and the last 3093 points in the series are used as test series. For instance, a training set of size 1000 involves training using $z(6000)$ through $z(7000)$. The employed nets have one external input and ten hidden units. For each number of increasing training set sizes, we train ten networks using different random initial weights and compute the resulting normalized mean squared error (NMSE) on the test set. NMSE is defined by

$$\text{NMSE} = \frac{|\mathcal{S}|^{-1} \sum_{t \in \mathcal{S}} e^2(t; \hat{\mathbf{w}})}{\widehat{\text{var}}(d(t))} \tag{10}$$

where t runs over the set $\mathcal{S}$ in question (i.e., either training or test set), $|\mathcal{S}|$ is the size of the set, and $\widehat{\text{var}}(\cdot)$ denotes the empirical variance.

The learning curve is shown in the right panel of Figure 1. Initially the test error drops as the size of the training set is increased, but from training set size 2500 to 5500 the average test error is fairly constant. This can be explained by visual inspection of the laser series as the "shape" of many collapses between the corresponding points 1500–4500 seems atypical for the test series. We see a significant drop in test error when increasing the training set size from 5500 to 6000 points which might be explained by the fact that the training set now incorporates an additional collapse very similar in shape to the ones in the test series. These observations suggest that for the laser series, the concept of an example should be conceived on several time scales: there are the pointwise examples corresponding to each single input presented to the network; but more important, there obviously exists "super examples" consisting of a whole section of the time series. If additional super examples or sections are not similar to the sections encountered in the test series, generalization will not improve as seen in the right panel of Figure 1.

We now examine the memory of selected networks. The left panel of Figure 2 depicts the normalized version of Eq. (7) for increasing values of lag space m when evaluating one of the networks with low test error trained on 7000 examples. The horizontal dotted line indicates the normalized level

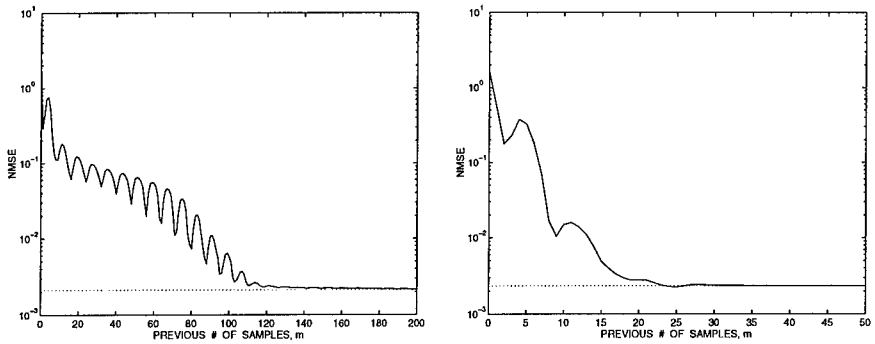


Figure 2: Left panel: Measuring *average* memory for one of the networks with low generalization error trained on 7000 examples from the laser series. Right panel: Measuring *average* memory for another of the networks trained on 7000 examples.

$\widehat{G}_\infty(\widehat{\mathbf{w}})$ using all available previous inputs. It seems that the network has a memory somewhere between 120 and 200. The precision ϵ in (8) denotes a level below which we consider the two errors as equivalent. The value of the memory thus naturally depends on the choice of ϵ as shown in Table 1. In the right panel of Figure 2 the normalized test error for increasing lag

ϵ	0.05	0.025	0.01
M	150	183	198

Table 1: The value of the memory dependence on ϵ for curve in the left panel of Figure 2.

space m for another of the nets trained on 7000 points is shown. We note that for this network the memory M is less sensitive to ϵ, as it is between 23–25 for $\epsilon \leq 0.18$. We also note that the memory is much shorter than for the previous network even though the test errors are almost identical. Note, since the network complexity[6] is restricted, a network with short memory is able to allow for more individual contribution of each of the previous inputs $x(t-n)$ than a network with long memory. The memory profile of a short term memory net is thus more fine grained than that of a long term memory net (with the same complexity). One might claim that a compact memory model is better tuned to the problem.

In the left panel of Figure 3 we illustrate the average memory of the network with lowest test error when training on only 500 examples. We notice that by limiting the memory the error can actually become lower than $\widehat{G}_\infty$. This effect often occurs for *overtrained* networks which is also the case here. The memory of the network is highly specialized on the training set; limiting the memory acts as regularization and actually improves the performance on the test set.

We now illustrate that the memory of a recurrent network indeed is a

[6]E.g., measured by the number of hidden neurons.

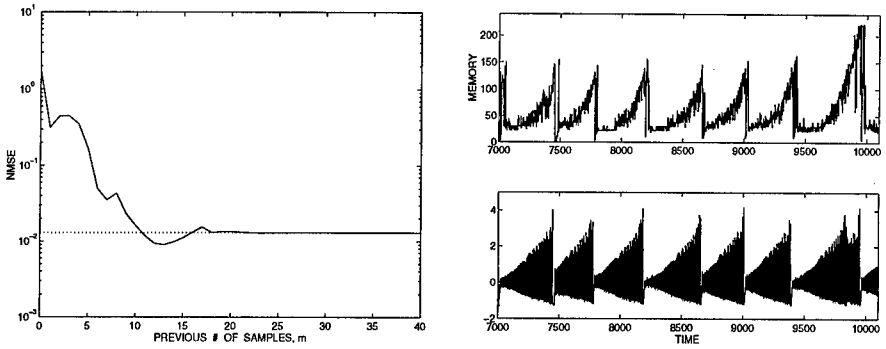


Figure 3: Left panel: Measuring *average* memory for best network trained on 500 examples from the laser series. Right panel: Measuring *local* memory with threshold $\epsilon = 0.01$ using five point average, $K = 5$.

dynamic quantity by examining the *local* memory defined by Eq. (8) and (9) for the network whose *average* memory is shown in the left panel of Figure 2. The right panel of Figure 3 and the left panel of Figure 4 illustrate the dynamic memory measure using precision $\epsilon = 0.01$ and averaging over $K = 5$ and $K = 50$ examples, respectively. The memory is seen to be very dependent upon where in the laser series it is measured; the closer to a collapse, the larger. The memory required around the last collapse is significantly larger than around the previous collapses. This may be explained by the observation that the characteristics of the laser series just before the last collapse is highly atypical from the rest of the test series. The memory in the right panel of Figure 3 averaging over only $K = 5$ previous errors is seen to be a very noisy quantity. As K is increased the error measure becomes smoother. Recall from Table 1 that the average memory for $\epsilon = 0.01$ is $M = 198$; however, the illustrations of the local memory shows that by omitting the last collapse the average memory would be measured to 150, approximately.

The Mackey-Glass series is a standard problem of nonlinear dynamics and results from the integration of a differential equation, see e.g., [8]. Standard

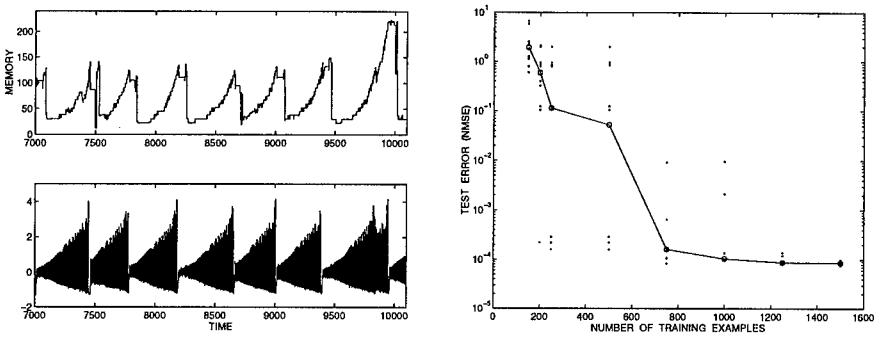


Figure 4: Left panel: Measuring *local* memory with threshold $\epsilon = 0.01$ using fifty point average, $K = 50$. Right panel: Learning curve for the Mackey-Glass series.

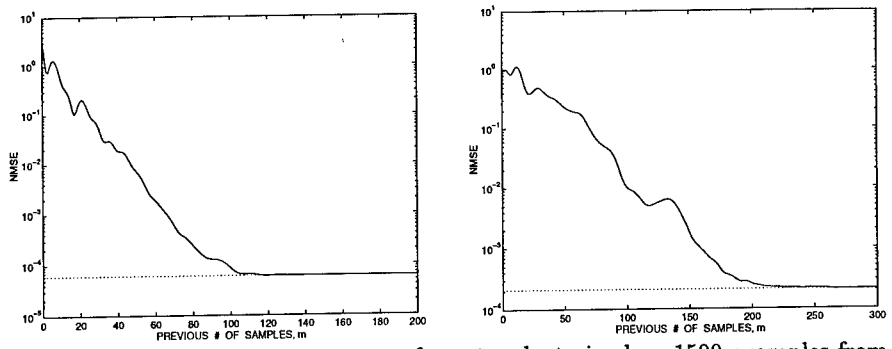


Figure 5: Measuring *average* memory for networks trained on 1500 examples from the Mackey-Glass series. Left Panel: Network having short memory. Right panel: Network having long memory.

practice is to implement a six step ahead predictor, i.e., modeling $z(t)$ from a lag space vector $\mathbf{x}(t) = [z(t-6), z(t-12), \ldots, z(t-6n_I)]$ using feed-forward networks. Here we implement the six step ahead predictor with target value $d(t) = z(t)$ using a recurrent network with only one external input, $x(t) = z(t-6)$, and ten hidden units. In the right panel of Figure 4 is shown a learning curve for the Mackey-Glass series when training on up to 1500 samples and testing on the following 7000 samples. For each training set size ten networks were trained. The learning curve indicates that more than 1000 examples are needed in order to obtain consistently good results on the test set. We then determined the average memory defined by Eq. (7) for the properly trained networks with the lowest errors on the test set. Using the threshold $\epsilon = 0.01$ we found that the networks implemented a memory in the range of 118–263, as seen from Figure 5.

The memories implemented by the recurrent networks are surprisingly long. In order to obtain comparable performance using feed-forward networks six external inputs are needed, thus spanning a total of only 31 previous samples. This is the minimal memory neccessary for good performance provided weighting of individual lags is possible, however, a RNN's memory profile is more coarse grained reducing the possibilty of individual weighting. Furthermore, maintaining information about all previous input values seems to bias recurrent networks towards the implementation of a long *effective* memory.

The long memory implemented by the recurrent networks seems to be of prime importance for the *robustness* of these models. Preliminary experiments indicate that recurrent networks are far more resilient to noise pertubations of the input data than comparable feed-forward networks. Examination of the robustness of recurrent networks is a topic of ongoing research.

CONCLUSION

In this paper we have focused on determining the *effective* memory of recurrent neural networks when used for time series processing, equivalent to

the span of the externally provided lag space for feed-forward networks. In particular, we have suggested an operational definition which measures the memory of a fully trained RNN on a test set. The viability of the method is illustrated on two chaotic time series problems.

ACKNOWLEDGMENTS

This research was supported by the Danish Natural Science and Technical Research Councils through the Computational Neural Network Center (CONNECT). JL furthermore acknowledge the Radio Parts Foundation for financial support. Lars Kai Hansen is acknowledged for stimulating discussions.

REFERENCES

[1] H.D.I. Abarbanel: **Analysis of Observed Chaotic Data**, New York, NY: Springer-Verlag, 1996.

[2] M. Casey: "The Dynamics of Discrete-Time Computation, with Application to Recurrent Neural Networks and Finite State Machine Extraction," **Neural Computation**, vol. 8, pp. 1135–1178, 1996.

[3] S. Haykin: **Neural Networks: A Comprehensive Foundation**, New York, New York: Macmillan College Publishing Company, 1994.

[4] T. Lin, B.G. Horne, P. Tino & C.L. Giles: "Learning Long-term Dependencies with NARX Recurrent Neural Networks," **IEEE Transactions on Neural Networks**, vol. 7, no. 6, p. 1329, 1996.

[5] M.W. Pedersen & L.K. Hansen: "Recurrent Networks: Second Order Properties and Pruning," in G. Tesauro, D. Touretzky & T. Leen (eds.) **Advances in Neural Information Processing Systems 7**, Cambridge, MA: The MIT Press, 1995, pp. 673–680.

[6] M.W. Pedersen: "Training Recurrent Networks," in **Proceedings of the IEEE Workshop on Neural Networks for Signal Processing VII**, Piscataway, New Jersey: IEEE, 1997.

[7] H.T. Siegelmann, B.G. Horne & C.L. Giles: "Computational Capabilities of Recurrent NARX Neural Networks," **Technical Report UMIACS-TR-95-12, IEEE Transactions on Systems, Man and Cybernetics**, 1997 (in press).

[8] C. Svarer, L. K. Hansen, J. Larsen & C. E. Rasmussen: "Designer Networks for Time Series Processing," in C. A. Kamm, G. M. Kuhn, B. Yoon, R. Chellappa & S. Y. Kung (eds.), **Proceedings of the IEEE Workshop on Neural Networks for Signal Processing 3**, Piscataway, New Jersey: IEEE, pp. 78–87, 1993.

[9] A.S. Weigend, & N.A. Gershenfeld (eds.): **Time Series Prediction: Forecasting the Future and Understanding the Past**, Santa Fe Institute Studies in the Sciences of Complexity, Reading, MA: Addison-Wesley, 1993.

EXTRACTING THE RELEVANT DELAYS IN TIME SERIES MODELLING

Cyril Goutte
Department of Mathematical Modeling - Bygn. 321
Technical University of Denmark, DK-2800 Lyngby, Denmark
Phone: +45 4525 3921
Fax: +45 4587 2599
E-mail: cg@imm.dtu.dk

Abstract. In this contribution, we suggest a convenient way to use generalisation error to extract the relevant delays from a time-varying process, i.e. the delays that lead to the best prediction performance. We design a generalisation-based algorithm that takes its inspiration from traditional variable selection, and more precisely stepwise forward selection. The method is compared to other forward selection schemes, as well as to a non-parametric tests aimed at estimating the embedding dimension of time series. The final application extends these results to the efficient estimation of FIR filters on some real data.

OVERVIEW

In system identification as well as in time series modelling, the choice of the inputs to our model plays a crucial role. In order to obtain good performance, one shall model future behaviour from a set of *relevant* past measurements. An insufficient amount of inputs will prevent the model from capturing the underlying mapping. On the other hand, including irrelevant inputs will lead to poor prediction performance, as suggested by the "curse of dimensionality".

In this contribution, we consider a method aimed at finding a set of relevant delays. For that purpose, we use a suboptimal iterative method that minimises the estimated generalisation error, and bears resemblance to the usual statistical variable selection methods [6]. However, this Extraction of Relevant Delays (ERD) method is original in the fact that 1) it assesses the relevance of possible inputs on the basis of generalisation, and 2) it is adapted to time dependant problems.

The organisation of this paper is as follows: first we give a short presentation of the topic of statistical variable selection, and describe our ERD method. We then introduce briefly a class of methods estimating the em-

0-7803-4256-9/97/$10.00

bedding dimension of time series. The second part of the paper contains a number of experiments conducted on the well-known Hénon map, on a real time series, and finally on a FIR filtering problem. We conclude with a discussion of the results.

INPUT SELECTION

Let us consider a standard time series modelling problem. A sequence x of measurements is collected, and we try to predict x_t from a set of past values x_{t-d}. Note that in that setting, the length of the basic time delay (i.e. difference between t and $t+1$) is imposed on us. Extracting the relevant delays consists in finding a set of m delays $(x_{t-d_1}, \ldots, x_{t-d_m})$ that, given as input to a model, yields the best prediction.

This is a special case of variable selection, which in turn can be seen as part of the more general problem of analysing the structure in the data [6]. An important assumption in conventional variable selection is that all necessary variables are available, i.e. a sufficient subset of inputs actually exists. Provided that data are sampled correctly, this assumption is usually satisfied in the case of time series[1]. We will use the terms 'variable', 'input' or 'delay' indifferently when addressing our time series modelling problem.

An exhaustive search through all possible subsets of inputs is usually impossible for combinatorial reasons. A number of suboptimal techniques have thus been designed, among them stepwise methods:

- *Forward selection* methods consists in starting from an empty set of inputs, and adding variables one after the other according to a given *selection criteria*, until a chosen *stopping condition* is fulfilled.
- On the contrary, *backward elimination* methods start with the full set of inputs, and proceed by deleting one variable at a time according to the *selection criteria*, until the *stopping condition* is reached. In the field of neural computation, variable selection techniques based on pruning [2] are a typical example of backward elimination.

Stepwise regression usually refers to a combination of both (in the linear case). For both methods, the crucial parts are the design of the selection criteria, and the stop condition. Conventional methods in linear regression rely on e.g. correlation coefficients, information content or F-testing.

EXTRACTION OF RELEVANT DELAYS

We present here a method of Extraction of Relevant Delays (ERD) that relies upon generalisation error. It draws its inspiration from forward selec-

[1] It breaks down in the case where a long-term delay is needed, that ranges further than the time period spanned by the data. However, the relevance of such long-term prediction is questionable, and there would be no data to identify the associated parameter(s) anyway.

tion methods, combined with generalisation estimation. Consider a model f providing a mapping from an input vector containing m delays $\mathbf{x}^{(t)} = (x_{t-d_i})_{i=1\ldots m}$ to output x_t, and assume Gaussian perturbation on the output. We define the *generalisation error* (or expected risk) for this model as:

$$G(f) = \int \left(f\left(\mathbf{x}^{(t)}\right) - x_t\right)^2 p\left(x_t, \mathbf{x}^{(t)}\right) \mathrm{d}x_t \mathrm{d}\mathbf{x}^{(t)} \tag{1}$$

Obviously, equation (1) can not be used directly as the joint input-output probability is unknown. We will thus resort to estimating this error, or rather its average over all possible training sets of a given size N. There are mainly two classes of such estimators: methods such as cross-validation [17] resample the available data, while algebraic estimators [1] rely on statistical arguments.

Consider for example the second option. Many estimators have been proposed in the literature, e.g. Final Prediction Error (FPE) [1], Generalised Prediction Error (GPE) [11], Final Prediction Error for Regularised problems (FPER) [7] or Network Information Criterion (NIC) [12]. We will here settle for an expression similar to GPE, i.e. a FPE where the number of parameters is replaced by the *number of efficient parameters* $\widehat{P}$:

$$\langle \widehat{G} \rangle = \left(\frac{N + \widehat{P}}{N - \widehat{P}}\right) \langle S \rangle \tag{2}$$

where $\langle S \rangle$ is the average training error (over all training sets of size N). As such an average is not available, we plug the measured training error (or empirical risk) $S(f)$ instead. For quadratic risk, $S(f) = \sum \left(f\left(\mathbf{x}^{(t)}\right) - x_t\right)^2$. The calculation of $\widehat{P}$ depends on the regularisation method used during training (see e.g. [7, 3]).

The proposed ERD method is a forward method taking all delays in their natural order (which bypasses the *selection criteria*), and adds a candidate input if and only if it corresponds to a *significant* decrease in generalisation error. The algorithm can be described as follows:

1. Initialise: $d = 0$; $G_{min} = \sigma_x^2$; no input selected.
2. Model: $d = d + 1$; add delay $t - d$ to selected inputs; estimate generalisation error $\widehat{G}$ for resulting model.
3. Test: if $\widehat{G}$ is significantly smaller than G_{min}, keep delay $t - d$; $G_{min} = \widehat{G}$. Discard otherwise.
4. Iterate: Go to step 2 until stop condition is reached.

Significant decrease in error. When a candidate delay yields a decrease in (estimated) generalisation error, step 3 requires that we assess the significance of this decrease. We take advantage of the fact that the generalisation estimators mentioned above are based on averaging a statistics, and test whether the statistics associated with two different generalisation estimators have statistically significantly distinct means by performing a paired t-test [15, 8].

In our case, the estimated (average) generalisation error given by FPE can be expressed as the following average:

$$\langle \widehat{G} \rangle = \frac{1}{N} \sum_{k=1}^{N} \left(\frac{N + \widehat{P}}{N - \widehat{P}} \right) e_w^{(k)} \tag{3}$$

where $e_w^{(k)}$ is the local risk (e.g. squared residuals) for training example k and a model parameterised by w. Let us consider two models trained on the same set of examples, and $\widehat{P}_1$ and $\widehat{P}_2$ the *numbers of efficient parameters* for the first and second model (respectively). The distribution of the corrected residuals $\left(\frac{N+\widehat{P}_1}{N-\widehat{P}_1}\right) e_{w_1}^{(k)}$ (resp. $\left(\frac{N+\widehat{P}_2}{N-\widehat{P}_2}\right) e_{w_2}^{(k)}$) has mean $\langle \widehat{G}_1 \rangle$ (resp. $\langle \widehat{G}_2 \rangle$). We thus test whether $\langle \widehat{G}_2 \rangle$ is significantly smaller than $\langle \widehat{G}_1 \rangle$ by using a paired t-test on the corrected residuals.

The case of cross-validation is somewhat more straightforward. The *leave-one-out* (LOO) cross-validation score is calculated by averaging the prediction error on one example for a model trained on the remaining sample:

$$\langle \widehat{G} \rangle = \frac{1}{N} \sum_{t=1}^{N} \left(f_t \left(\mathbf{x}^{(t)} \right) - x_t \right)^2 \tag{4}$$

Where f_t is the model trained *without* example $\left(\mathbf{x}^{(t)}, x_t\right)$. For two different models, the residuals are paired according to the example left out, so that a (paired) t-test can be used to determine whether these residuals come from distribution with different mean, i.e. correspond to different average generalisation errror. Extension to *m-fold* cross-validation is straightforward.

EMBEDDING DIMENSION

In the study of non-linear dynamical systems, and time series in particular, an important problem lies in finding the *embedding dimension* [16], which is essentially equivalent to finding the set of *primary* delays in time series. In the realm of neural computation, the recently proposed δ-test method[14] addresses this issue. In a different field, a method for identifying the order of non-linear input-output systems was proposed [5], that relies on the use of "Lipschitz quotients" i.e. ratio between output and input distances. A similar method applied to time series (called 'geometrical technique') was presented last year at this workshop [10].

Though different in practice, these methods rely on a common assumption on the continuity of the underlying mapping, and use a geometrical approach based on the data alone. The continuity argument means that if there is a mapping between $\mathbf{x}^{(t)}$ and x_t, then close inputs $\mathbf{x}^{(u)}$ and $\mathbf{x}^{(v)}$ should correspond to close outputs x_y and x_u. Accordingly, as long as the input space is insufficient (i.e. missing delays), close inputs can correspond to arbitrarily distant outputs. Quantifying this is done either by measuring empirical probabilities that two outputs are close given that the corresponding inputs are

close (δ-test), or by calculating the ratio between output and input distances (Lipschitz quotients).

It should be noted that these methods are non-parametric. They rely on the data alone, and need not specify a given model (contrary to the ERD method). This can turn out to be a disadvantage since for a given data set, they always select the same set of relevant delays, regardless of the ability of our model to actually implement the underlying mapping. It could very well be that for the model at hand, the estimation would benefit from the inclusion of a *secondary* delay, as shown in the next section and discussed further down. Furthermore, these geometrical techniques require extensive calculations, as they consider all pairs of data. They are thus computationally expensive.

TIME SERIES EXPERIMENTS

This section is devoted to two simple experiments. First we use an artificial problem (the Hénon map), for which a large validation set confirms the results obtained by our ERD method. In the second experiment, we discover interesting long term dependencies on a real time series.

The Hénon map is implemented by the following mapping: $x_t = 1 - 1.4x_{t-1}^2 + 0.3x_{t-2}$. We generate a training set containing 500 data, and a test set of 10000 elements for assessing generalisation abilities. We experiment on non-noisy as well as noisy data, with $\sigma_\varepsilon^2 = 0.1$. Two different models are used: a linear model (obviously ill-suited to this purpose) and a non-parametric kernel smoother. The generalisation estimators are the FPE and LOO respectively.

In order to check whether the delays are wisely chosen, experiments are performed comparing the ERD method and other selection methods (table 1):

1. a forward selection methods using a large validation set (distinct from the test set) of 10000 data;
2. the F_{99}-inclusion, a selection scheme based on the F-statistics [6];
3. the δ-test [14].

As shown on table 1, all forward selection methods outperform the δ-test in the linear case: a linear combination of the first two delays is obviously insufficient to model the mapping. The performance is rather homogeneous among forward selection methods, though the ERD method tends to favour parsimonious models, while keeping good generalisation abilities.

On the non-noisy data, the kernel smoother captures the underlying mapping in all cases. When the training data is noisy, the F-inclusion scheme displays a severe case of curse of dimensionality. The other methods select

Hénon map:		No noise		Noisy	
		Linear	Kernel	Linear	Kernel
Large validation set	Delays	1–7	1–2	1–7	1–3
	MSE	0.376	0.000	0.503	0.214
	Generalisation	0.379	0.000	0.389	0.067
ERD	Delays	1,3–6	1–2	1,3,4	1–3
	MSE	0.376	0.000	0.523	0.214
	Generalisation	0.378	0.000	0.409	0.067
F_{99}-inclusion	Delays	1–6	1–2	1–6,10	1–8,11–13, 16,17,19,20
	MSE	0.376	0.000	0.499	0.032
	Generalisation	0.379	0.000	0.389	0.294
δ-test	Delays	1–2		1–2	
	MSE	0.457	0.000	0.567	0.266
	Gener.	0.455	0.000	0.459	0.097

Table 1: Results on the noisy and non-noisy Hénon map data, for two models: a linear model and a non parametric Kernel smoother. MSE is the Mean Squared (training) Error, generalisation is estimated on 10000 non-noisy data.

one additional delay $t-3$. As we will discuss later, this theoretically unnecessary input leads to an improved prediction accuracy on both the training and generalisation set.

Fraser river data. As an example of real time series processing, we will use a publicly available dataset[2] containing the mean monthly flow of the Fraser River in Hope, British Columbia, from march 1913 to December 1990 [9]. It is a roughly periodic data set containing 946 measurements with maxima every 11 to 13 months. We split the data set so that we have half the data for training and half for testing the prediction abilities of the model. In the following experiments, we use the log values of the data, and estimate the parameters by minimising the Mean Squared Error on the transformed data.

The use of a large validation set is not possible here as is (unfortunately) the case with most real life problem. We will compare the result of the ERD scheme to the results provided by the non-parametric δ-test. According to this test, the embedding space of the time series involves 6 delays.

Note that the ERD method once again outperforms the method based on estimating the embedding dimension. The linear model probes further into the past, and spots relevant delays up to $t-48$, i.e. four times the time span covered by the δ-test. The kernel smoother seems to be experiencing some problems coping with the dimensionality of the data—they could probably be minimised using a variable metric. The neural networks model selects the same amount of delays than the δ-test. However, the selection is targeted towards minimisation of generalisation error, which is reflected in the sizeably smaller test error. Noticeably, the non-linear neural network model, though

[2]available on Statlib at `http://lib.stat.cmu.edu/datasets/`

Fraser river :		Linear	Kernel	Neural net
ERD	Delays	1,2,4–7,10,11, 23,26,35,48	1,2,4,7,11,13	1,2,4,7,11,23
	MSE	0.0529	0.0389	0.0425
	Generalisation	0.0439	0.0547	0.488
δ-test [14]	Delays	1,2,4,7,8,11		
	MSE	0.0680	0.0441	0.452
	Generalisation	0.0609	0.0530	0.627

Table 2: Results for the Fraser river data set, and three different models. MSE is the Mean Squared Error.

using a combination of regularised cost optimisation and OBS pruning [4], does not manage to extract longer-term delay, and is outperformed by the simpler linear model.

OPTIMISING FIR FILTERS

We will now extend the method and apply it to fMRI signal modelling. The fMRI signal measures the hemodynamic response to focal neuronal activation. The data is collected as a 504 steps time-series containing measurements corresponding to the hemodynamic response to a series of periodic baseline and activation periods (7 periods in all). The data is corrupted by a very high level of noise.

Modelling this response as a function of the activation signal is the object of active current research [13]. We extend the above method to optimise the choice of relevant delays when trying to model the response with a FIR filter applied on the excitation signal. Current attempts at doing so use a fixed lag of 7 delays.

The ERD method is simply extended by testing sequentially chosen delays in the excitation signal rather than the time series itself. We applied the method on 5 voxels that were identified as being particularly responsive to the excitation. Out of the 504 measurements, we set the last two periods, or 144 data, aside for testing the generalisation abilities. The first 5 periods, containing 360 points, are used for identifying the relevant delay and the filter coefficients. The FPE is used as a generalisation estimate.

On the 5 fMRI time series studied, we extracted from 1 to 4 delays, ranging from $t-1$ to $t-22$. On voxel number 3 for example, our experiments surprinsingly select only $t-1$, but we can see on figure 1 (left panel) that this actually leads to a slight decrease in generalisation error compared to the fixed 7 delay filter. Overall, the results displayed on the left panel of figure 1 suggest that on the extremely noisy data, the method leads to performance that is comparable to the fixed FIR filter, while using less parameters.

On the first voxel, the extraction of relevant delays leads to a noticeable decrease in generalisation error. The right panel of figure 1 plots the response of voxel 1 together with both FIR estimation.

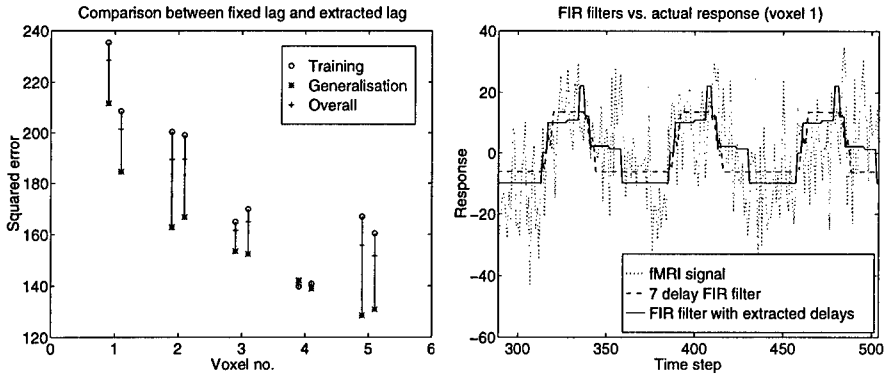


Figure 1: Left: performance comparison. For each voxel the 7 delays FIR filter is on the left, while the FIR filter with extracted delays is on the right. Right: behaviour of the 7 delay filter and the extracted filter on the fMRI time series measured in the first voxel.

DISCUSSION

The FIR example above illustrates the fact that using a parcimonious model, with delays appropriately chosen, is a good way of obtaining good modelling abilities. This can be of great help when facing a problem on which we have no—or little—physical insight. In that context, the ERD is a principled model-dependent approach that has the ability to select the inputs that lead to the best expected prediction error.

It should also be emphasised that it seeks to optimise the actual criteria of interest, i.e. generalisation error. Indeed, at the end of the day we are interested in obtaining the best possible predictions. Reconstructing the dynamics of a time series, as suggested by the methods aimed at estimating the embedding dimension, is only a way of reaching this ultimate goal. On the contrary, the ERD method that we present here tries to optimise the relevant performance criterion directly.

This has an interesting effect: by essence, the ERD method takes into account the fact that modelling is performed on a limited amount of data. On the Hénon map example, this leads to the selection of an additional delay. It has no link to the actual dynamics of the system, but gives a clear decrease in error. Furthermore, when the model is not flexible enough to implement the system mapping, we will probe further into the past, and possibly discover higher-order dependencies that will ease the modelling. This is well illustrated by the two time series examples.

Another aspect of the delay extraction procedure as proposed here is that it relies on the estimation of the generalisation error. It is expected that the more accurate the prediction is, the more relevant the delays selected will be. It should be noted however that we are only interested in finding minima of the generalisation error, so an estimator will be usefull as long as it gives the right "trend" in generalisation.

Lastly, let us recall that this method is inspired from the *forward selection*

methods in statistical variable selection. A natural extension of this is the use of *backward elimination* steps, in a manner similar to stepwise regression. Similarly, pruning techniques can be used to remove inputs of the model that are potentially harmfull with respect to generalisation error.

SUMMARY

We have presented a generalisation-based method for finding the relevant delays in time series modelling. It relates to stepwise variable selection procedures in classical (linear) regression. This 'Extraction of Relevant Delays' method is straightforward to implement and leads to interesting results. When the model is not flexible enough to implement the underlying mapping, it selects additional delays in order to minimise estimated generalisation performance. Noticeably, it outperforms some non-parametric methods for determining the embedding dimension when applied to insufficiently flexible models.

Directions for future work include refinement of the relevance criterion, as well as the extension of this scheme to different problems such as system identification, with more than one temporal inputs.

Acknowledgements. I am very grateful to Jan Larsen for extremely valuable discussions on variable selection. This work was supported by a research fellowship from the Technical University of Denmark.

REFERENCES

[1] H. Akaike, "Fitting autoregressive models for prediction," **Annals of the Institute of Statistical Mathematics**, vol. 21, pp. 243–247, 1969.

[2] T. Cibas, F. Fogelman Soulié, P. Gallinari and S. Raudys, "Variable selection with Optimal Cell Damage," in **Proceedings of ICANN'94**, 1994, pp. 727–730.

[3] C. Goutte, "On the use of a pruning prior for neural networks," in **Neural Networks for Signal Processing VI – Proceedings of the 1996 IEEE Workshop**, 1996, pp. 52–61.

[4] B. Hassibi and D. G. Stork, "Second order derivatives for network pruning: Optimal brain surgeon," in S. Hanson, J. Cowan and C. Giles, eds., **Advances in Neural Information Processing Systems**, Morgan Kaufmann, 1993, vol. 5 of **NIPS**, pp. 164–171.

[5] X. He and H. Asada, "A new method for identifying orders of input-output models for nonlinear dynamic systems," in **American Conference on Control**, San Francisco, California, 1993.

[6] R. R. Hocking, "The analysis and selection of variables in linear regression," **Biometrics**, vol. 32, pp. 1–49, March 1976.

[7] J. Larsen and L. K. Hansen, "Generalized performance of regularized neural networks models," in **Neural Networks for Signal Processing IV – Proceedings of the 1994 IEEE Workshop**, 1994, pp. 42–51.

[8] J. Larsen and L. K. Hansen, "Empirical generalization assessment of neural network models," in **Neural Networks for Signal Processing V – Proceedings of the 1995 IEEE Workshop**, 1995, pp. 42–51.

[9] A. I. McLeod, "Diagnostic checking of periodic autoregression models with application," **Journal of Time Series Analysis**, vol. 15, no. 2, pp. 221–233, 1994.

[10] C. Molina, N. Sampson, W. J. Fitzgerald and M. Niranjan, "Geometrical techniques for finding the embedding dimension of time series," in **Neural Networks for Signal Processing VI – Proceedings of the 1996 IEEE Workshop**, 1996, pp. 161–169.

[11] J. Moody, "Note on generalization, regularization and architecture selection in nonlinear learning systems," in **Proceedings of the first IEEE Workshop on Neural Networks for Signal Processing**, 1991, pp. 1–10.

[12] N. Murata, S. Yoshizawa and S. Amari, "Network Information Criterion—determining the number of hidden units for an artificial neural network model," **IEEE Transactions on Neural Networks**, vol. 5, no. 6, pp. 865–872, 1994.

[13] F. Å. Nielsen, L. K. Hansen, P. Toft, C. Goutte, N. Lange, S. C. Strother, N. Mørch, C. Svarer, R. Savoy, B. Rosen, E. Rostrup and P. Born, "Comparison of two convolution models for fMRI time series," in **3rd Intl. Conference on Functional Mapping of the Human Brain**, 1997.

[14] H. Pi and C. Peterson, "Finding the embedding dimension and variable dependences in time series," **Neural Computation**, vol. 6, no. 3, pp. 509–520, May 1994.

[15] W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery, **Numerical Recipes in C**, Cambridge University Press, 2nd edn., 1992.

[16] R. Savit and M. Green, "Time series and dependent variables," **Physica D**, vol. 50, no. 1, pp. 95–116, 1991.

[17] G. Toussaint, "Bibliography on estimation of misclassification," **IEEE Transactions on Information Theory**, vol. 20, no. 4, pp. 472–479, July 1974.

COMBINED LEARNING AND USE FOR CLASSIFICATION AND REGRESSION MODELS *

David J. Miller, Hasan S. Uyar, and Lian Yan
Department of Electrical Engineering
The Pennsylvania State University
Room 227-C EE West Bldg.
University Park, PA 16802-2701
Phone: (814) 865-6510
E-mail: *miller@perseus.ee.psu.edu*

Abstract

We show that the decision function of a radial basis function (RBF) classifier is equivalent in form to the Bayes-optimal discriminant associated with a special kind of mixture-based statistical model. The relevant mixture model is a type of "mixture of experts" model for which class labels, like continuous-valued features, are assumed to have been generated randomly, conditional on the mixture component of origin. The new interpretation shows that RBF classifiers do effectively assume a probability model which, moreover, is easily determined given the designed RBF. This interpretation also suggests a maximum likelihood learning objective, as an alternative to standard methods, for designing the RBF-equivalent models. This statistical objective is especially useful for incorporating unlabelled data within learning to enhance performance. While this approach might appear to be limited to applications involving a large, label-deficient training set, the scope of application is significantly extended with the observation that any new data to classify is also *unlabelled data*, available for learning. Thus, we suggest a *combined learning and use* paradigm, to be invoked whenever there is new data to classify. This new approach is tested for vowel recognition, given a small archive of examples from different speakers. For this problem, a conventional method is of necessity speaker-independent. By contrast, combined learning and use allows speaker-dependent adaptation, with resulting gains in performance.

*This work was supported in part by National Science Foundation Career Award IRI-9624870.

0-7803-4256-9/97/$10.00 ©1997 IEEE

1 Combined Learning and Use

In the problem of statistical classification, there is generally a clear division between the design phase of the classifier and the phases in which the classifier is validated on test data, or in which the design is used in an application setting to classify new data. Typically, one is given a training set of representative (feature vector, class label) training pairs. This set is all that is available for the model design, and once the design is complete, the classifier is fixed for subsequent test set validation or for *use* in an application.

It is interesting to contrast this picture with learning in other domains, e.g. adaptive filtering methods in signal processing, where there is no clear division between *learning* and *use* since both occur simultaneously. A second example with analogies to classification is the problem of image segmentation. Here, one has a general model for images, in the form of an energy function that depends on both continuous-valued model parameters and on discrete segmentation (classification) variables. The energy function is minimized for each new image one wishes to segment. Each such minimization determines both the continuous model parameters *and* the segmentation for the image. Thus, *learning* of the model and its *use* (segmentation of the given image at hand) are combined into one operation.

There are compelling reasons for considering whether this combined learning and use analogy can be extended to the classification problem. In particular, in principle we would like to modify the classifier for new data (*data to be classified*) if the training data was an inadequate representation of the underlying data source, or if the data source is non-stationary. The primary difficulty with this objective lies in the fact that nearly all learning approaches for statistical classification are *pure* supervised learning schemes, for which a supervising datum, i.e. the class label, is required for each feature vector used in the training. This statement is clearly true of neural network learning approaches such as back propagation, which minimize the squared distance to target class values. It is also true of standard methods for training parametric (e.g. Gaussian mixture) classifiers, wherein the data is first divided into classes, with maximum likelihood estimation then applied to separately estimate the class-conditional densities. Unfortunately, new data to classify is inherently unlabeled, which makes it incompatible with standard supervised learning approaches. Moreover, even given the existence of a learning method which utilizes unlabeled data, it is not obvious that use of this data would be effective, without accompanying labels, in reducing the classification error rate. In fact, there are results which suggest otherwise [1].

However, recently, several new schemes have been suggested for assimilat-

ing unlabeled data within the learning. These methods have been found to be effective when the amount of labeled training data is inadequate. In [11], a method was proposed for training classifiers based on mixed labeled/unlabeled training sets. The authors suggested a statistical model for the data naturally suited to a maximum likelihood learning scheme involving both the labeled *and* unlabeled data subsets. The unlabeled subset essentially allows improved estimation of the model parameters, which are then "plugged into" the Bayes decision rule. In [8] and [7], we built on the previous work in [11], suggesting improvements to the classifier structure and to the learning method, and also introducing the concept of "combined learning and use" for classification. First, we suggested a more powerful probability model than that in [11] with an associated classifier structure that, in this paper, we show to be *equivalent* to the radial basis function (RBF) classifier [9]. Unlike standard RBFs, the RBF-equivalent probability model is amenable to likelihood-based training, which is the key to assimilating mixed labeled/unlabeled training data within the learning. Second, in [8] we proposed an alternative learning criterion to that in [11], based on the joint data likelihood over both the labeled and unlabeled data subsets[1], which was found to yield performance gains over the method in [11]. Two *distinct* (Expectation-Maximization) EM [3] learning algorithms were derived for maximizing this likelihood. These distinct methods were obtained by viewing different data elements as "missing data" within the EM framework [7]. Finally, we made the observation that test data, or in fact any new data to classify, can be viewed as a new unlabeled data set to which the mixed training method can be applied, to *modify* the classifier prior to the classifier's *use* on this data. This is what we called "combined learning and use".

In this work, we first briefly summarize the developments in [7], after which we show the equivalence between the probability model introduced there and the RBF classifier. Next, we validate the combined learning and use paradigm for this classifier structure through an experiment involving vowel recognition given an archive of examples from different speakers. In this context, given a limited training set, a conventional classification approach is of necessity *speaker-independent*. By contrast, we will show that combined learning and use provides a way to achieve *speaker-dependent* adaptation (and use) of the model, with resulting gains in the classifier performance. Finally, we suggest that the combined learning and use approach may also be applicable to regression fitting.

[1] A conditional data likelihood measure was suggested in [11].

2 Mixed Training for an RBF-equivalent Mixture Model

Although the classifier learning problem has been separately cast as one of i) directly estimating a posteriori class probabilities, ii) least squares regression to target class values [2], and iii) more directly minimizing an error count measure [6], only the first objective appears suitable for incorporating unlabeled data within the learning. Moreover, while there are neural network models which directly produce the class probability estimates as outputs[3], the training for such networks is also unsuitable for incorporating unlabeled data. Alternatively, in [7], we suggested a somewhat unconventional probability model. This model is a generalization of a standard mixture wherein, like the feature vectors $x \in \mathcal{R}^k$, the class labels $c \in \mathcal{I}$, $\mathcal{I}$ the label set, are also assumed to have been generated *randomly*, conditional on the mixture component of origin. More concretely, the data is assumed to have been generated in the following way:

1. Randomly select one of M mixture components according to the probability mass function $\{\alpha_l, l = 1, \ldots, M\}$.
2. Given the selected component, k, choose: a) a feature vector x according to the component density $f(x; \Lambda_k)$, where Λ_k is the parameter set of the density, and b) a class label c according to the conditional probabilities $\{\beta_{j|k}, j = 1, \ldots, |\mathcal{I}|\}$.

Note that usually mixture components are "hard-partitioned" to classes, i.e. $\beta_{j|k} \in \{0, 1\}$. However, we have found that allowing classes to probabilistically "share" components makes the learning less sensitive to initialization and the solution less sensitive to the choice of M. This model will also be motivated from a different standpoint shortly.

The corresponding a posteriori class probabilities take the form:

$$P[c|x] = \sum_k \beta_{c|k} \left(\frac{\alpha_k f(x; \Lambda_k)}{\sum_l \alpha_l f(x; \Lambda_l)} \right). \quad (1)$$

These probabilities have a "mixture of experts" structure [4], where the "gating units" are the conditional probabilities of mixture components given feature vectors (in parentheses), and with the "expert" for component k just the

[2] This objective is also related to estimating the a posteriori probabilities [10].

[3] MLP structures with normalization in the output layer can be trained as probability estimators by minimizing a cross entropy criterion.

conditional probability $\{\beta_{c|k}\}$.The associated Bayes decision rule is

$$S_{\text{bayes}}(x) = \arg\max_j \frac{\sum_k \alpha_k \beta_{j|k} f(x;\Lambda_k)}{\sum_k \alpha_k f(x;\Lambda_k)}, \tag{2}$$

where $S_{\text{bayes}}(\cdot)$ is a selector function with range in $\mathcal{I}$.

For the learning problem, we supposed the existence of two data subsets, i.e. $\mathcal{X} = \{\mathcal{X}_l, \mathcal{X}_u\}$, where $\mathcal{X}_l = \{(x_1, c_1), (x_2, c_2), \ldots, (x_{N_l}, c_{N_l})\}$ is the labelled subset, with $\mathcal{X}_u = \{x_{N_l+1}, \ldots, x_N\}$ the unlabeled one. Unlabeled data was introduced into the learning to improve parameter estimates via the joint data likelihood criterion:

$$\log L = \sum_{x_i \in \mathcal{X}_u} \log \sum_{l=1}^{M} \alpha_l f(x_i;\Lambda_l) + \sum_{x_i \in \mathcal{X}_l} \log \sum_{l=1}^{M} \alpha_l \beta_{c_i|l} f(x_i;\Lambda_l). \tag{3}$$

A description of two distinct EM learning algorithms for maximizing $\log L$ can be found in [7]. Finally, it was recognized that any new data set to classify can be viewed as a new subset $\mathcal{X}_u$ – hence the data to classify can be used to update the model (e.g. to specialize the model) prior to the model's use in classifying the data – *i.e.*, a combined learning and use operation was suggested.

Note that mixed training and "combined learning and use" only appear to be applicable to classifier structures that are based on a probability model for the data. However, we now show that this assumption is not very restrictive, since the decision rule for the probability model we just described is actually equivalent in form to a commonly used neural network classifier that is not typically given a probabilistic description. Consider a radial basis function network used in a classification setting [9]. For an $|\mathcal{I}|$-class problem, there is one RBF output per class,

$$g_j(x) = \sum_{k=1}^{M} f(x;\Lambda_k)\lambda_{kj}, \quad j = 1, \ldots, |\mathcal{I}|. \tag{4}$$

Here, $f(\cdot)$ is the ***basis function*** which, without much restriction, we may take to be a density function. Also, λ_{kj} is a scalar weight connecting basis k and class output j [4]. The associated decision function, $S : \mathcal{R}^m \rightarrow \{1, 2, \ldots, |\mathcal{I}|\}$, is the winner-take-all rule:

$$S(x) = \arg\max_j g_j(x). \tag{5}$$

[4]In [9], an alternative ***normalized*** RBF structure was suggested. For convenience we consider the un-normalized structure here, though our equivalence result holds for both structures because the normalization does not affect the decision rule.

We can observe that $g_j(\cdot)$ looks like a mixture density. However, it is not usually thought of in this way, primarily because the weights $\{\lambda_{kj}\}$ can be positive or negative and are trained for a least squares criterion [9], rather than a statistical one such as the data likelihood. Regardless, we can easily show that the decision rule (5) is in fact equivalent to the Bayes rule in (2). First, define $\lambda_{\min} = \min_{k,j} \lambda_{kj}$. Then, subtracting $(\lambda_{\min} \sum_k f(x;\Lambda_k))$ from each class output, we obtain the equivalent rule:

$$S(x) = \arg\max_j \sum_k f(x;\Lambda_k)\tilde{\lambda}_{kj}, \tag{6}$$

where $\tilde{\lambda}_{kj} \equiv \lambda_{kj} - \lambda_{\min}$. Note that $\tilde{\lambda}_{kj} \geq 0$. Next, we divide each output by the constant $(\sum_{m,n} \tilde{\lambda}_{mn})$, to obtain

$$S(x) = \arg\max_j \sum_k f(x;\Lambda_k)q_{kj}, \tag{7}$$

where $q_{kj} \equiv \frac{\tilde{\lambda}_{kj}}{\sum_{m,n} \tilde{\lambda}_{mn}}$. Now, we have $0 \leq q_{kj} \leq 1$ and $\sum_{kj} q_{kj} = 1$. Finally, we can normalize each transformed class output $\tilde{g}_j(x) \equiv \sum_k f(x;\Lambda_k)q_{kj}$ by the sum $\sum_{l=1}^{|\mathcal{I}|} \tilde{g}_l(x)$ to again yield the rule

$$S(x) = \arg\max_j \frac{\sum_k f(x;\Lambda_k)q_{kj}}{\sum_{m,n} f(x;\Lambda_m)q_{mn}} = \arg\max_j \frac{\sum_k f(x;\Lambda_k)(\frac{q_{kj}}{\sum_n q_{kn}})(\sum_n q_{kn})}{\sum_m f(x;\Lambda_m)(\sum_n q_{mn})}. \tag{8}$$

Now, comparing (2) with (8), it is easy to see that they are equivalent. In particular, we may identify $\frac{q_{kj}}{\sum_n q_{kn}}$ with $\beta_{j|k}$ and $\sum_n q_{kn}$ with α_k. There are two implications of this result. First, we suggest an alternative probabilistic viewpoint on RBF classifiers. In fact, given a standard RBF solution, we have shown that one can easily find the equivalent probability model. While the parameters of this equivalent model were not obtained via maximum likelihood estimation, the equivalent model may still provide some insight into the *implicit* statistical modelling assumptions made by the RBF solution. Moreover, in some cases one is interested both in hard classification decisions *and* in a probabilistic assessment of class ownership. The RBF-equivalent model directly provides this information via the probabilities $\{P[c|x]\}$. The second implication of this equivalence, and the one of more significance for

this work, is that it effectively suggests new training possibilities for a widely used neural network classifier. In particular, while the standard RBF model is not amenable to likelihood-based training (and hence to combined learning and use), the RBF-equivalent model is. We next consider one scenario where combined learning and use, applied to this RBF-equivalent model, yields performance benefits. Other results can be found in [7].

3 Experimental Results

In a basic application of the combined learning and use paradigm, we simply use the new unlabeled features associated with the test set to augment an existing training set. After training on the combined data set using the scheme in [7], the new data set is classified. As one example in [7], we considered the 3-class, 40-dimensional, 5000 sample *waveform-+noise.data* set from the UC Irvine machine learning repository. There, it was shown that combined learning and use outperformed conventional approaches for training RBF classifiers, which are forced either to discard the unlabelled features, or to make limited use of them. Here, we apply this new paradigm to the classification of a speech *archive*. The archive includes examples of vowels from different speakers, with the speaker identity known. While examples of the same vowel may share similar statistics across speakers, examples of *different* vowels from the *same* speaker may also share a common statistical character. This suggests that it may be sensible to separate the data into "speaker-dependent" batches. Each such batch to classify can be taken as an unlabeled set for combined learning and use, either in concert with a labeled training set (derived either from the entire archive, or from an independent data set) or to modify a previous design based on such a labeled training set. The success of this scheme rests with the potential for adapting the classifier to each data batch, based on the unlabeled batch features.

We have tested this idea on Deterding's *vowel.dat* set, consisting of LPC-derived log area ratios, representing eleven different vowels. In this set, there are six examples of each vowel from each of fifteen different speakers (990 examples in all). We used two examples of each vowel from each speaker as the labeled training set (330 samples in all) with the remaining four examples from each speaker used as the test set. Note that the data set is too small to design separate classifiers for each speaker in a conventional way. We chose to compare two different combined learning approaches, along with the method from [9] (denoted MD-RBF). In the first scheme (*speaker-independent* (SI)), the test set of 660 samples was viewed collectively as $\mathcal{X}_u$ and used in concert with $\mathcal{X}_l$ for combined learning. We then classified $\mathcal{X}_u$. In the second

method (*speaker-dependent* (SD)), the forty-four test set samples from each speaker were viewed as distinct data batches. We thus performed combined learning separately for each speaker, based on $\mathcal{X}_l$ and the speaker-specific test set. Each batch was then classified based on its speaker-specific model. Note the tradeoff between the two designs: the speaker-specific scheme designs each of the fifteen classifiers using only the 330 labeled samples and 44 speaker-specific unlabeled samples, while the speaker-independent scheme uses the entire data set for its single design. However, this latter classifier must perform well for all speakers, rather than just a single one. Results were obtained for models of size 11, 22, and 33 mixture components. The performance measure chosen was the average test set error fraction, computed based on different choices for the parameter initialization and for the data subset realization. We designed classifiers for all possible (fifteen) realizations of the training and test sets. For each realization, for each model size, 20 classifiers were designed based on random parameter initialization, with the test set performance averaged over the 300 solutions. As shown in Table 1, the speaker-specific scheme provides a consistent performance advantage over the speaker-independent one[5], with both methods outperforming MD-RBF.

Model Size	Method	Train-err/Test-err
11 components	SD	0.58/0.58
	SI	0.61/0.61
	MD-RBF	0.63/0.64
22 components	SD	0.45/0.46
	SI	0.49/0.50
	MD-RBF	0.50/0.52
33 components	SD	0.37/0.38
	SI	0.41/0.42
	MD-RBF	0.44/0.44

Table 1: Average test set misclassification error fraction for Deterding's *vowel.dat* set. Results for two "combined learning and use approaches" (speaker-independent (SI)) and (speaker-dependent (SD)) are shown, along with MD-RBF.

[5]The high error rates observed for this experiment are consistent with prior results reported for this data set.

4 Combined Learning and Regression Modelling

There is a natural way to extend the combined learning and use approach developed heretofore to address the "other" basic supervised learning problem – regression. For this problem, the data pairs now take the form (x_i, y_i), where $y_i \in \mathcal{R}^m$ is the regression *output* with x_i now called the *input*. Combined learning and regression fitting can be accomplished by modifying existing learning approaches that are already likelihood-based, to incorporate an additional unlabeled term. Jacobs and Jordan [4] and Jordan et al. [5] suggested "mixture of experts" regression structures naturally suited for learning procedures based on maximum likelihood estimation. We can pose the learning problem as maximizing the conditional likelihood $\log f(\mathcal{Y}|\mathcal{X})$ as in [5], or the joint likelihood $\log f(\mathcal{Y}, \mathcal{X})$ as in [12]. For combined learning and use application, we suppose that there is now an "unlabeled" new/test input data set, for which we want to estimate the outputs. Using "l" to denote the "labeled" subset, we can now alternatively maximize either $\log f(\mathcal{Y}_l|\mathcal{X}_l) + \log f(\mathcal{X}_u)$ or $\log f(\mathcal{Y}_l, \mathcal{X}_l) + \log f(\mathcal{X}_u)$. We are currently investigating combined learning and regression as outlined here. It is important to note that combined learning and use may have a smaller range of application for regression than for classification. In particular, for certain regression problems such as time series prediction, the actual outputs are made available as time unfolds – hence, this problem lacks a large, output-deficient set on which to apply combined learning and use. One likely application, however, is the problem of restoring images observed through a noisy, distorting medium. Cha and Kassam [2] used a model and learning approach similar to that in [4] and [5], with the learning performed over a training set of distorted images. We believe that combined learning and regression may be effective in this setting.

References

[1] V. Castelli and T. M. Cover. On the exponential value of labeled samples. *Pattern Recognition Letters*, 16:105–111, 1995.

[2] I. Cha and S. A. Kassam. RBFN restoration of nonlinearly degraded images. *IEEE Trans. on Image Processing*, 5:964–975, 1996.

[3] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum-likelihood from incomplete data via the EM algorithm. *Journal of the Roy. Stat. Soc., Ser. B*, 39:1–38, 1977.

[4] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Comp.*, 3:79–87, 1991.

[5] M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Comp.*, 6:181–214, 1994.

[6] B.H. Juang and S. Katagiri. Discriminative learning for minimum error classification. *IEEE Trans. on Sig. Proc.*, 40:3043–3054, 1992.

[7] D. Miller and H. S. Uyar. A mixture of experts classifier with learning based on both labelled and unlabelled data. (To appear in the proceedings of *Neural Information Processing Systems*, 1997.).

[8] D. Miller and H. S. Uyar. A generalized Gaussian mixture classifier with learning based on both labelled and unlabelled data. In *Proceedings of the Conference on Information Sciences and Systems.*, 1996.

[9] J. Moody and C. J. Darken. Fast learning in locally-tuned processing units. *Neural Comp.*, 1:281–294, 1989.

[10] M. D. Richard and R. P. Lippmann. Neural network classifiers estimate Bayesian a posteriori probabilities. *Neural Comp.*, 3:461–483, 1991.

[11] B. Shashahani and D. Landgrebe. The effect of unlabeled samples in reducing the small sample size problem and mitigating the Hughes phenomenon. *IEEE Trans. on Geoscience and Remote Sensing*, 32:1087–1095, 1994.

[12] L. Xu, M. I. Jordan, and G. E. Hinton. An alternative model for mixtures of experts. In *Neural Information Processing Systems*, volume 7, pages 633–640, 1995.

REDUCING FALSE ALARM RISK IN TRANSIENT SIGNAL CLASSIFICATION

de **LASSUS H** (*), **LECACHEUX A** (*), **DAIGREMONT P** (**),
BADRAN F(**), **THIRIA S** (***),
(*)Laboratoire ARPEGES, CNRS URA 1757,
Observatoire de Paris-Meudon - 92195 MEUDON cedex, France
(**)CEDRIC, Conservatoire National des Arts et Métiers,
292 rue Saint Martin - 75003 PARIS, France
(***)Laboratoire d'Océanographie et de Climatologie (LODYC),
Université de Paris VI, 4 place Jussieu, T14 - 75005 PARIS,
E-mail:delassus@mesioq.obspm.fr

Abstract - We address the problem of autonomous decision making in classification of radioastronomy transient signals on spectrograms from spacecraft. It is known [10] that the assessment of the decision process can be divided into acceptation of the classification, instant rejection of the current signal classification, or rejection of the entire classifier model. We propose to combine prediction and classification with a double architecture of Neural Networks to optimize a decision while minimizing the false alarm risk. We suggest a method to derive the input and output windows for the predictor network. Results on real data from URAP experiment aboard Ulysses spacecraft show that this scheme is tractable and effective. Keywords: spectrogram classification, radioastronomy, neural network classification.

1 INTRODUCTION

As transient astronomical signal detection, classification, and tracking from space is becoming a classic application for neural networks [4], in the literature though, measuring the risk of classification or detection decision is not always taken into account as it is for radar or sonar. Classification risk minimization has been studied by Bishop [2] and Schurmann [6], they suggest a Bayesian analysis of confidence, while Lippmann [7] uses a two networks approach based on bootstrapping to predict the risk of classification. However, low frequency radio planetary emissions are non stationary, non Gaussian and non linear.

0-7803-4256-9/97/$10.00

For such signals, classification risk assessment is not obvious.

A solution has recently been suggested by de LASSUS & al. [5] with two neural networks working in parallel, thus, only smooth signals were considered. We propose here a more general approach enabling false alarm risk reduction whatever the signal time frequency distribution, provided that the signal is correlated on the time frequency plane to some extent. The solution we suggest is based on neural classification of the signal confirmed by a neural prediction. With this method, it is possible to punctually reject the classifier's choice, or even, if necessary, to decide that the current classifier becomes unreliable and has to be changed. The example we show belongs to low frequency radio astronomy, but this approach seems applicable in many other fields of interest where geophysical signals are under study.

2 CLASSIFICATION OF SPECTROGRAMS

Classification of low frequency planetary radio bursts displayed in the time frequency plane is a task similar to the well known "cocktail party" problem: identify many sources emitting in the same time. The signals under study originate from different parts of the solar system, and the distances, as well as the observation angles, are heterogeneous and changing fast. This leads to moving patterns on the spectrogram while we want to classify the signal. Automatic tracking and classification of non Gaussian, nonlinear transient radio planetary signal on spectrogram, using a single detector is a difficult task. When the omni directional detector is moving fast across the sources and when the patterns have an anisotropic distribution along the trajectory of the detector, abrupt changes occur on the spectrogram. An accurate classification may be cast into more simple subtasks : identify the sources number, find the main features of the sources, and classify the signal. If the classification has to be done on spectrogram representation of the signal, the Time Delay Neural Network (TDNN) architecture has proved to be a convenient choice [4]. A classifier would typically have a 20×12 input, a 1×3 output, and 1240 parameters for 9609 learning samples. On each frequency channel, a TDNN classifier is moved from one energy peak of the signal to another. An input window is cut on the spectrogram around the peak. The TDNN output gives the label (class of signal) of the current energy peak on the spectrogram. Typically, the method yields a rate of success of up to 90%, challenging human expert visual recognition. Thus, since the number of sources may change, as well as their patterns, the classifier may be confronted to situations unseen before. In such cases, there is a need to assess the quality of the decision, and to evaluate the ability of the classifier to carry on its mission successfully.

The rejection dilemma. Let d_1 be the decision of accepting a classifier's suggestion, and d_0 the rejection of this proposal. Let H_1 be the hypothesis

that the signal belongs to the label suggested by the classifier, and H_0 hypothesis that it does not belong to this label. Then, we have [10]: choose d_1 when H_1 (correct classification); choose d_1 when H_0 (false alarm); choose d_0 when H_1 (undue rejection); choose d_0 when H_0 (justified rejection). If the TDNN output number is strictly designed according to the number of known classes of signal, the risk of d_0H_0 and d_0H_1 is limited to the set of outputs of the neural network, and the classifier is forced to decide between the available classes [8]. This is a good solution when the number of classes is computable by direction finding together with clustering on the cumulant domain, or on a wavelet packet base [3]. But even so, we still have to deal with d_1H_0 . In order to reduce the risk of d_1H_0 without loosing too much on d_1H_1, one would intuitively try to use different thresholds on the output of the classifier, but this is inappropriate if the probability densities of each signal class in the learning data base are equal. The solution that we suggest is rejection by prediction: that is to predict the signal outside of the TDNN classifier input. If this short term prediction is confirmed by future samples, then the postponed decision is confirmed.

3 REJECTION BY PREDICTION

Principles. In order to minimize the classification risk d_1H_0, we need a measure of quality to motivate our decision. We suggest to use two neural networks (NN) in parallel, a classifier and a predictor, for each frequency channel. Inputs to these NNs will be subsets of the same window cut on the spectrogram. One NN will classify the signal with the information available from the window. The other NN will predict the signal expected to come just outside the window in the very near future. When the predicted signal is there, it is confronted with the prediction. The distance from the prediction to the real signal yields the measure of quality we needed to confirm the decision of the classifier. In order to cover a sufficient area of the time frequency plane, we decided to predict the future signal at five different locations (time frequency coefficients). This number of five is not fundamental and can be modified. We show now how positive autocorrelation lags yield the customized architecture for the predictor NN we need.

Designing a predictor. The problem is to choose the right window and the suitable time step for the prediction. It is known [1] that the optimal input for our classifier is determined in the ambiguity plane by the first zero crossings of the 2D-autocorrelation function of the signal spectrogram. The input window of a classifier TDNN (noted I_c in the following) has been optimized in order to include all types of signal (wide band smooth, wide band bursty, and band limited signals respectively noted S_1, S_2, S_3). We want for each class S_i $i = 1 \ldots 3$ to determine a smaller window (I_p) included in I_c window, and five points O_i $i = 1 \ldots 5$ outside I_p where prediction will be

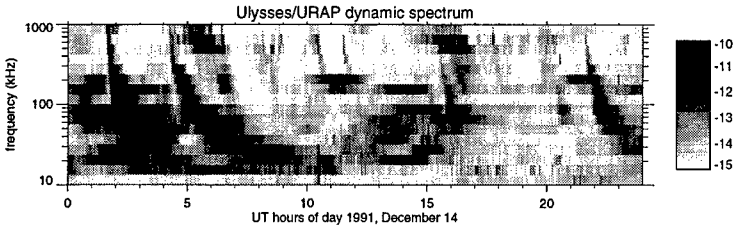
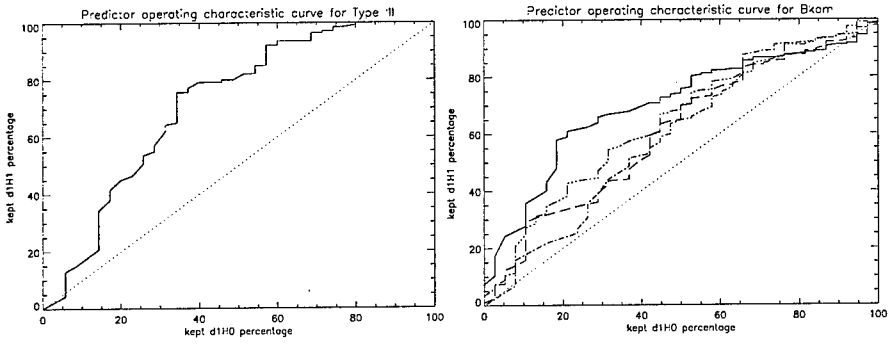


Figure 3: Typical Ulysses Spectrogram with 4 different types of signals overlapping

Data Analysis. For our experiment with real data, we used eight months of data obtained by the URAP experiment aboard Ulysses spacecraft [9] (September 1991 - April 1992). In these data, radio bursts from the Sun and Jupiter are continuously present. Since Ulysses flew by Jupiter on Feb 8^{th}, 1992 and changed its trajectory plane by Jupiter gravity assist, the morphology of the highly directive radio emissions from Jupiter changed abruptly. Radio spectra were acquired every 144 sec; they are made of 16 frequency channels logarithmically spaced from 10 to 1000 kHz. Signal intensities were normalized between (-1,+1). From September to February, four kinds of radio emissions are present, two smooth signals (Solar Type III, Jovian nKOM) and two bursty signals (Jovian hOM and bKOM), then from February the 8^{th} 1992 until April, the two bursty signals (Jovian hOM and bKOM) disappear abruptly and a new bursty signal is present (Kom). The two smooth signals are present on the entire data set.

Experiment. We selected a learning set of 9609 events from September to December 1991, and a test set of 486 events collected during these months, but not the same days. The test set was used to stop learning early enough to keep the generalization properties of the NNs. A validation set of 951 events was then selected from January to February 8^{th}, 1992. We tried our method on the 100kHz frequency channel, which lies at the center of the spectrogram, because there, the task was most difficult with four overlapping signals. In fact, any other frequency could have been chosen. For the Type III signal, the constructive algorithm gave a window 9×2 and five points in the neighborhood of the window. So we derived an MLP with 9×2 inputs, 5 outputs, and 222 parameters. We predicted the signal at the five chosen locations where autocorrelation was maximum outside the input window. In the last layer of the multi layer perceptron, we omitted the sigmoidal mapping, so that garbage detection criteria can easily be derived from observing the values of the outputs [6]. A threshold is calculated from statistics (μ and σ) of the prediction error on the learning data set. A classified sample is rejected when its prediction error is higher than the threshold. Similarly we derived a predictor MLP for the nKOM signal with an input window 7×2, and 5 outputs. For the bKOM signal the best window was very small, 2×2 so we derived an MLP with a 2×2 input, 5 outputs and 30 parameters.



(a) (b)

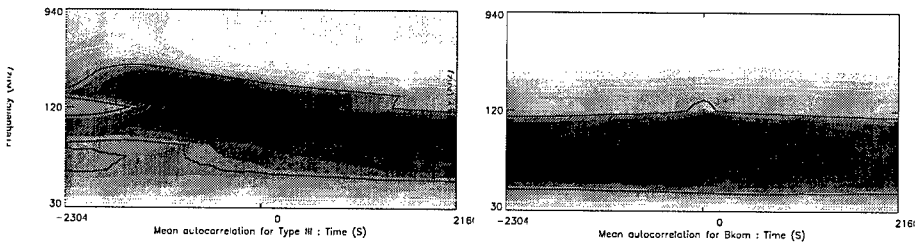
Figure 4: Percentage of d_1H_0 risk reduction (X axis) against percentage of d_1H_1 loss (Y axis). (a) Type III, (b) bKOM.

Results and Discussion. The two plots in figure 4 show the predictor operating characteristic curve for two different types of signals, plotting d_1H_0 risk percentage against d_1H_1 risk percentage. The nearer the curve approaches the upper left corner of the figure, the better the results. Performance can be computed by integration of this curve. On figure 4 (b) the first four curves of the bKOM list are plotted. It is noticeable that the constructive algorithm gives us the windows in an ordered list of decreasing efficiency. The lower performance on nKOM (not shown on the figures) has an explanation. nKOM signal is seldom seen and the training database does not contain as many nKOMs as needed to reach the same standard as other signals. These results indicate that point wise rejection of d_1H_0 is possible provided that the input window is optimized for the signal under study. Moreover it demonstrates that peak correlation is a key factor for prediction. We will see now another interesting result: the possibility of model rejection when a new signal is received.

4 MODEL REJECTION

Principles. The idea is to use the MLP predictors and the TDNN classifier jointly, to assess the ability of the classifier to pursue its task. It means that if the signals present in the learning set have disappeared, and have been replaced by new signals, learning has to be resumed, and the classifier changed. A constrained classifier alone gives no information on its ability to carry on its task. If the prediction error of the successive current MLP predictor is kept in memory, then its standard deviation, calculated on a moving window, gives information on the presence of a wave form that was not present in the learning data set. A suitable error bar calculated from the learning data will tell if the classifier has to be changed.

made. Theoretical results [1] imply that the size of I_p has to be different for each class S_i, depending upon the stationarity of the signal measured by its autocorrelation. Figures 1(a)(b) show the different autocorrelations for two types of signal: a smooth signal (solar Type III), and a bursty signal (Jovian bKOM). Classification and prediction will only be possible where isolines show positive autocorrelation, i.e. on dark areas, which are varying from one signal to another. The classifier's input can be centered on the energy peak, the size of the window being adjusted to the mean zero crossing. This choice is not possible for the predictor because some room has to be left for the predicted output somewhere on the highest part of the autocorrelation function. This leads to an iterative optimization process to choose the best compromise between input and output settings and sizes.



(a) (b)

Figure 1: Time frequency autocorrelation of two signals. Dark areas show strong correlation with the signal energy peak. (a) Type III, (b) bKOM.

Specification for the windows. Considering the type of signals we were studying we decided to predict its behavior on five different locations chosen outside of the input window where autocorrelation is at its highest level. Doing so we can be assured that prediction is done precisely where the signal is locally quasi stationary, so that our chance of success is maximum. The input window must be rectangular to be suitable for the TDNN classifier. According to our experiments, it seems that TDNNs networks are more efficient than MLP for classification while the latter is better for prediction. We found also that MLP is somewhat more sensible to noise than TDNN, when predicting. However, we assigned a multi layer perceptron (MLP) for the prediction task. As a specific predictor NN has to be derived for each type of signal, we suggest a general method to choose the input and output layers of the predictor NN.

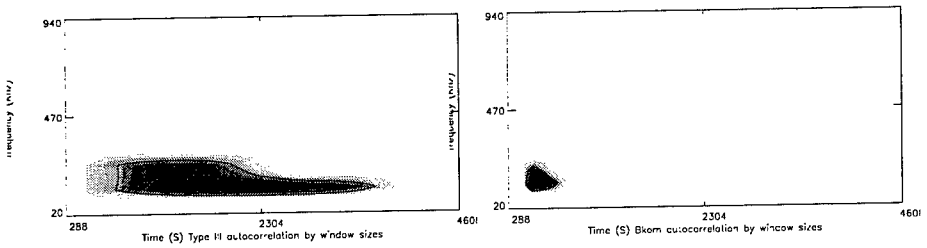
Derivation of window size and location. The general idea is to adjust the input layer as close as possible to the unknown matched filter. Set a small candidate input window (I_p) (e.g. of size 2×2) for the predictor over the energy peak at a given frequency. Compute C_k (correlation between the

window and the peak for signal S_k) :

$$C_k = \frac{1}{N\sigma_{I_p}} \sum_{j=1}^{N} \sum_{i=1}^{M} (x_j - \bar{x_{I_p}})(x_i - \bar{x_{I_p}})$$

where σ_{I_p} is the mean standard deviation of I_p windows, N is the number of samples in the learning database of the same signal S_k, M is the size of the window, x_j is the peak. $\bar{x_{I_p}}$ is the mean energy of the window around peak x_j. Try all possible positions for this window around the peak. Enlarge the window and iterate the process as long as C_k is positive. Select the window which maximizes C_k computed for the k class over the learning database, and then determine the related location of the desired prediction (points O_i $i = 1 \ldots 5$). Choose these points outside of the input window, where correlation is highest. The output of the predictor NN will have to predict these points (one output per point to predict). The result can be seen on figure 2, for two types of signal. It gives a mapping of autocorrelation of all possible windows from size 1×1 to size 32×16. The areas in grey levels show efficient window sizes. The best window is given by the darkest point of the figure. Y size is read vertically, x size is read horizontally. For smooth wide band signals (Solar Type III), good windows are 2500 seconds long and 2 frequency channels high. For bursty wide band signals (Jovian bKOM) efficient windows can be found in a small area around 300 seconds large and two frequency channels high. Whereas for band limited signals (Jovian nKOM not represented here), relevant windows can be found along a narrow band of two frequency channels of 2000 seconds long. With our data base, convergence was almost immediate.

The next step is to learn the prediction, test the predictor with a test set, and choose a rejection error bar according to the performances on the test set. Eventually validating the system on a validation set.



(a) (b)

Figure 2: Mapping of I_p window autocorrelation according to its size. Window time width on X axis, frequency height on Y axis. Dark areas show strong correlations. (a) Type III, (b) bKOM.

Experiment. For this experiment we chose the data from September 1991 to April 1992. On the 8^{th} of February the bKOM/hOM disappears and is replaced by Kom, and from this date, the classifier is unable to operate. This

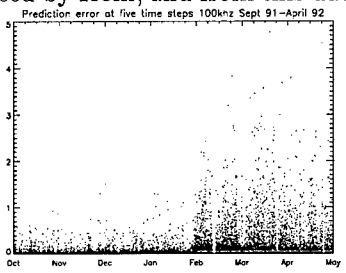


Figure 5: Prediction error

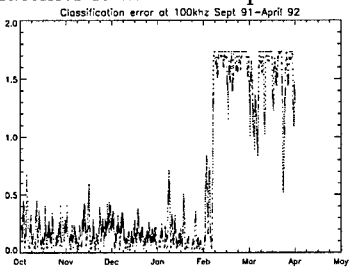


Figure 6: Classification error

is obvious from the peak starting from February on figure 6 which illustrates classification error calculated for the period. Figure 5 shows the prediction error during the same period. The appearance of the new signal is accurately detected within a few minutes. Spurious peaks after February indicate clearly that the classifier is obsolete. Detailed analysis of the results show that the valleys in the prediction error are correlated with those of the classification error. They correspond to the presence of recognized signals (Type III). These signals are jammed by a new Kom signal never seen in the learning data base.

5 CONCLUSION

We have proposed a method to manage rejection in classification of spectrogram of low frequency radio astronomy signals. We have shown that a predictor MLP and a classifier TDNN can be used in parallel to minimize the false alarm risk. Moreover, this scheme can be used to decide if the classifier has to be changed when the environment has evolved. This capacity is of crucial importance for space radioastronomy detectors which have no directivity. We have shown how to derive the best input and output windows for the predictor neural network. This method gives us a series of windows of decreasing efficiency and enhances the fact that ambiguity plane correlation is a key factor for transient Bayesian prediction.

REFERENCES

[1] ALTES R. A. Detection estimation and classification with spectrograms. *Journal of the Acoustical Society of America*, 67(4), April 1980.

[2] BISHOP. *Neural Networks for pattern Recognition.* Oxford Press, Oxford, Grande Bretagne, 1995.

[3] P. H. CARTER. Unknown transient detection using wavelets. In Harold H. Szu, editor, *Wavelet Applications*, volume 2242, pages 803–814. Proc. SPIE the international Society for Optical Engineering, March 1994.

[4] H. DE LASSUS A. LECACHEUX S. THIRIA F. BADRAN. Neural Network Clusters and Cellular Automata for the Detection and Classification of Overlapping Transient Signals on Radio Astronomy Spectrograms from Spacecraft. In *International Symposium on Time-Frequency and Time-Scale Analysis*, pages 253–256, Paris, France, june 1996. IEEE Signal Processing Society.

[5] H. DE LASSUS P. DAIGREMONT A. LECACHEUX S. THIRIA F. BADRAN. Rejection by prediction. In *ICANN'96*, Bochum, Germany, july 1996.

[6] SCHURMANN J. *Pattern classification.* Wiley Interscience, 1996.

[7] LIPPMANN R. P. and al. Predicting the risk of complications in coronary artery bypass operations using neural networks. In *NIPS-7*, pages 1055–1062, MIT, Cambridge, USA, 1995.

[8] SEBASTYEN G. S.. *Decision Making Processes in Pattern Recognition.* Mac Millan, New York, USA, 1962.

[9] STONE and al. The Unified Radio and Plasma Waves Investigation. *Astron. Astrophys. Suppl. Series*, 92:291–316, 1992.

[10] H.L. VAN TREES. *Detection, estimation and modulation theory: part I 1968, part II 1971, part III 1971.* Wiley, New York, 1968.

MULTIPLE AND TIME-VARYING DYNAMIC MODELLING CAPABILITIES OF RECURRENT NEURAL NETWORKS

Andrew D. Back
Brain Information Processing Group, Frontier Research Program
The Institute of Physical and Chemical Research (RIKEN)
2-1 Hirosawa, Wako-shi, Saitama 351-01 Japan

Abstract

In this paper, we propose some theories regarding the dynamical system representational capabilities of recurrent neural networks with real-valued inputs and outputs. It is shown that multiple nonlinear dynamic systems can be approximated within a single nonlinear model structure. A relationship is identified between this class of recurrent network, hybrid models and agent based systems.

1 Introduction

Recurrent neural networks are very general models and have been proven to offer significant computational capabilities such as Turing equivalence with linear slowdown [18]. There has been some interest in applying recurrent networks to dynamical systems and control problems. It has been shown that such models possess universal dynamic approximation capabilities [9, 19].

Recently, it has been shown by Feldkamp and Puskorius [8], that a class of recurrent networks can learn to model several dynamical systems and switch between them, depending on the characteristics of the input signal. Instead of learning to model just one system, the network was able to learn several models with very different properties.

This type of model phenomenon is not unique to the above example however. There appear to be a number of interrelated methods in the literature which have been studied mostly independently, see for example: [1, 2, 4, 10–12, 16, 17].

In this paper, we examine this phenomema further, and propose a theory which explains how recurrent neural networks can possess the capability of modelling a number of dynamical systems simultaneously. Examples are also given.

0-7803-4256-9/97/$10.00

2 Preliminaries

Function approximation is the task of approximating a mapping given by the function $F_1 : \mathcal{R}^m \rightarrow \mathcal{R}^n$, where $\mathcal{R}$ is the usual Euclidean space. *Functional approximation* is the task of approximating a mapping $F_2 : \mathcal{C}(K) \rightarrow \mathcal{R}$ where $\mathcal{C}(K)$ is a Banach space of continuous functions on a compact set K, defined by the norm $\|f\|_{\mathcal{C}(K)} = \max_{x \in K} |f(x)|$. The implication here, is that we have a mapping F_2 which maps the past inputs[1] to the current output (i.e. a variable $\zeta \in \mathcal{R}$) [3]. *Operator approximation* is the task of approxiating a mapping $F_3 : \mathcal{C}(K) \rightarrow \mathcal{C}(K)$. Thus, in this case, we seek to approximate a mapping of an input sequence $x(t) \in \mathcal{C}(K)$ to an output sequence $y(t) \in \mathcal{C}(K)$.

We term $\mathcal{C}_o(K_o)$ the space of operators on a compact set K_o. The mapping from one operator space $\mathcal{C}_o(K_o)$ to another operator space $\mathcal{C}_o(K_o)$ is called, for lack of a better name, an *operational map.*

Definition 1 *An operational map H is defined by*

$$H : \mathcal{C}_o(K_o) \rightarrow \mathcal{C}_o(K_o) \tag{1}$$

Definition 2 *A recurrent network (RNN) is defined by*

$$\mathbf{x}(t+1) = \mathbf{f}(\mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{E}\mathbf{y}(t)) \tag{2}$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t) \tag{3}$$

where $\mathbf{x}(t) = [x(t)x(t-1)...x(t-n_x)]^T$ is an $n(n_x+1) \times 1$ vector, $\mathbf{u}(t) = [u(t)u(t-1)...u(t-n_u)]^T$, $\mathbf{y}(t-1) = [y(t-1)...y(t-n_y)]^T$, $\mathbf{f}$ is $n(n_x+1) \times 1$ vector-valued function with sigmoid elements, typically defined as $\{f(x)\} = \tanh(x)$, $\mathbf{A} \in \mathcal{R}^{n(n_x+1) \times n(n_x+1)}$, $\mathbf{B} \in \mathcal{R}^{n(n_x+1) \times mn_u}$, $\mathbf{C} \in \mathcal{R}^{pn_y \times n(n_x+1)}$, $\mathbf{D} \in \mathcal{R}^{pn_y \times m(n_u+1)}$ and $\mathbf{E} \in \mathcal{R}^{n(n_x+1) \times pn_y}$. The bias terms are not explicitly shown, but are included within $\mathbf{u}(t)$ as a fixed-value input.

This characterization of a recurrent network gives a general framework from which many well known structures can be derived, e.g. [18–20].

Definition 3 *A* multiple model $\mathcal{M}_1(\mathcal{S}_1, \theta_1)$ *with fixed structure $\mathcal{S}_1$ and parameter vector θ_1 exhibits a set of characteristic properties $\mathcal{P}_1'$, where*

$$\mathcal{P}_1' = \begin{cases} \mathcal{P}_{1i} & i = 1, 2, ... \quad \textit{discrete multiple model} \\ \mathcal{P}_{1v} & v \in \mathcal{R}. \quad \textit{continuous multiple model} \end{cases} \tag{4}$$

3 Multiple Model Representational Using Operational Maps

A multiple model G can be characterized as an operator, which itself, consists of multiple operators F. Consider a multiple model operator $\Lambda : x \rightarrow y$, $x \in$

[1] A sequence $x(t)$, which we may sample at discrete points $t = 0, 1, ...$ is given by $x_s(t) = [x(0), x(1), \cdots]^t$ and is a function in $\mathcal{C}(K)$.

$X \subseteq \mathcal{C}(K)$, $y \in Y \subseteq \mathcal{C}(K)$ describing the input-output functional relationship. Additionally, Λ contains an operator F subject to, for example, $\Xi : F_a \rightarrow F_b$ $F_a \in \mathfrak{F}_a \subseteq \mathcal{C}_o(K_o)$, $F_b \in \mathfrak{F}_b \subseteq \mathcal{C}_o(K_o)$. Now, instead of considering Λ directly, we are interested in the existence of, and mechanisms by which there may arise mappings of the form $H : F_i \rightarrow F_{i+1}$, where $i = 1, 2, ...$ is the index of the discrete functionals in the discrete multiple model case, or $H : F_a \rightarrow F_b$ corresponding to the continuous multiple model case. In each case, $H \subseteq \mathcal{C}_\Omega(K_o)$ where $\mathcal{C}_\Omega(K_o)$ is the space of operational maps.

Hence, in addressing the issue of multiple models, this implies that we seek to answer the following question: 'Is it possible to find a nonlinear model which approximates an operational map $H : \mathcal{C}_o(K_o) \rightarrow \mathcal{C}_o(K_o)$?' We consider this question below.

The existence of a universal general operational map is made clear by the following theorem.

Theorem 1 *An operational map H given by*

$$F_\theta, x(t) \longmapsto H(F) : x(t) \rightarrow y(t) \tag{5}$$

can be obtained by the interconnection of a parameterized operator $F_\theta : \mathcal{C}(K) \rightarrow \mathcal{C}(K)$, and functional $M_c : \mathcal{C}(K) \rightarrow \mathcal{R}^m$ according to

$$\begin{aligned} y &= F_\theta(x;\theta) && (6)\\ \theta &= M_c(x) \qquad \theta = \theta_0\ t \leq 0 && (7) \end{aligned}$$

where $x(t)$, $y(t) \in \mathcal{C}(K)$ are continuous, real-valued input and output functions respectively of $t \in \mathcal{R}_+$, θ is the m-dimensional parameter vector of F_θ and θ_0 is the initial parameter vector.

Proof Sketch. Let there exist an operator F_θ determined by the parameter vector θ and a functional map M_c capable of universal approximation in the sense of , for example, [15]. For the ith parameter within a given model F_θ, we have

$$s_i = \theta_i z_i \tag{8}$$

Now, let M_c be a functional. Therefore, for any input sequence $x \in X$, M_c results in any desired set of parameters $\tilde{\theta} \in \tilde{\Phi} \subseteq \mathcal{R}^m$, such that

$$\begin{aligned} s_i &= M_{ci}(x) z_i && (9)\\ &= \tilde{\theta} z_i && (10) \end{aligned}$$

where $M_c = [M_{c1} \cdots M_{cm}]^T$ is a vector function. Thus, the existence of $M_c : u \rightarrow \tilde{\theta}$ permits any arbitrary F_θ to be obtained due to the mapping M_c. The operational map $H = \{F_\theta, M_c\}$ is general in the sense that it is capable of the same approximation as F_θ, but can be varied arbitrarily for any sequence x.

4 Universal Operational Maps

From Theorem 1, we can obtain the following result.

Theorem 2 (Universal Operational Map-I) *There exists a parameterized model* $G(\theta) : x(t) \to \hat{y}(t)$, *where* $x \in X \subseteq \mathcal{C}_o(K_o)$ *and* $\hat{y} \in Y \subseteq \mathcal{C}_o(K_o)$, *such that*

$$|y(t) - \hat{y}(t)| < \epsilon \tag{11}$$

where $F_{\theta'}, x(t) \longmapsto H(F) : x(t) \to y(t)$ *and* $\epsilon > 0$.

Proof Sketch. Let $F_{\theta'}$ be a parameterized model capable of universally approximating any operator (e.g. a Chen network [7]). Let M_c be a time delay neural network or other structure having universal functional approximation characteristics [6, 15]. Then from Theorem 1, there exists a neural network G defined collectively by $F_{\theta'}$ and M_c, which can approximate, arbitrarily closely, some operational map $H = \{F_{\theta'}, M_c\}$, where $H : \mathcal{C}_o(K_o) \to \mathcal{C}_o(K_o)$.

Remarks

1. The implication of the theorem is that every weight in F is replaced by an additional network M_{ci}. This provides the means of approximating nonlinear operational maps. As noted earlier, related approaches have been considered in the literature (see, for example, [16, 17]).
2. It is possible to introduce any required type of model for M_c. Thus, a hybrid model can be elegantly obtained.
3. The model G is a sigma-pi network, but can also be interpreted as a modular structure.
4. A more general form for M_c is $s = M_{ci}(x, u)$. In this case, the output from M_c is not used as a parameter, but receives the previous parameter input u and gives the previous output s after the parameter. Hence, we can derive the following related theorem.

Theorem 3 (Universal Operational Map-II) *A universal operational mapping* $F_\theta, x(t) \longmapsto H(F) : x(t) \to y(t)$ *is given by the interconnection of a universal operator* $F : (X, V) \to Y$ *and a single-input single-output universal function map* $M_{c1} : X \to V$ *according to*

$$y = F(x, v; \theta) \tag{12}$$

$$v = M_{c1}(x) \tag{13}$$

where $x \in X \subseteq \mathcal{C}(K), v \in V \subseteq \mathcal{C}(K), y \in Y \subseteq \mathcal{C}(K)$, *and* $\theta \in \Theta \subseteq \mathcal{R}^m$.

Proof Sketch. The proof follows directly from Theorem 1. The operator $F(x, v)$ is a universal approximator as independent as required in each of its inputs. Therefore, for every distinct value of v, e.g. $v = 1, 2, \ldots$ a distinct universal approximation $F : \mathcal{C}(K) \to \mathcal{C}(K)$ may be obtained. Let M_{c1} prescribe some operator from the input x to the extra input v. Therefore, a distinct universal approximation F can be obtained as required for any given input x, hence a universal operational model is obtained.

Remarks

1. In the above theorems, for universal operational maps, it may be assumed that F is given by a network as described in [7].

2. F may also be given by a recurrent neural network, with the universal approximation properties in the sense of those shown by Sontag [19]. In this case, the resulting recurrent network H, possesses also multiple modelling capabilities[2].

These results offer a different viewpoint for modular networks. Previously, modular neural networks were used to approximate a mapping by spatial decomposition [13]. Here, operational maps perform a temporal decomposition.

The discrete multiple model can be interpreted as a hybrid system [5]. Moreover, a neural network may arbitrarily form such hybrid systems, even in the course of training, *without the user necessarily being aware of this phenomena occurring.*

Discrete multiple models are related to agent systems [2, 14]. The models F and M_c can each be viewed as agents cooperating together to produce a more complex mapping than either is capable of acting alone. The framework proposed here also includes models such as mixtures of experts [10].

5 Synthesis of Multiple Models by Bias Shifting

Here we present a simple constructive approach to show how multiple models may be synthesized in both feedforward and recurrent neural networks. The approach we propose is well suited for multiple models which are comprised of a set of discrete F models and is applicable to synthesis as well as analysis, though the latter case is not discussed here.

Theorem 4 *An MLP model $G(x, v)$ can form multiple unique function mappings*

$$G(x, v) = m_v(x), \ v = 1, ...p \tag{14}$$

defined by

$$m_v(x) = \sum_{i=1}^{N} c_{vi} g \left(\sum_{j=1}^{m} \xi_{vij} x + \phi_{vi} \right) \tag{15}$$

where the extra input v indexes the desired mapping $m_v(x)$.

[2]Though to be precise, one may wish to qualify the sense of the approximation in terms of the specific characteristics of universal approximation being performed, i.e. in the sense defined by Chen and Chen [7] or Sontag [19].

Proof Sketch. Without loss of generality, let $N = np$. This specifies p subgroups within G, each of n hidden units. Therefore we have

$$G(x,v) = \sum_{i=1}^{N} c_{hi} g\left(\sum_{j=1}^{m} \xi_{hij} x + \theta_{hi} + rv\right) \quad h = 1,...p \tag{16}$$

$$\theta_{hi} = \phi_{hi} - rh \tag{17}$$

For sufficiently large r, if we consider approximations on the range $[a,b]$ where $|a,b| \ll r$, then setting $v = \{1,..,p\}$ results in

$$G(x,v) = \sum_{i=1}^{N} c_{vi} g\left(\sum_{j=1}^{m} \xi_{vij} x + \phi_{vi}\right) \tag{18}$$

$$= m_v(x) \tag{19}$$

as required.

Theorem 5 *An RNN model $G(x,y,v)$ can form multiple unique mappings*

$$G(x,y,v) = m_v(x,y), \; v = 1,...p \tag{20}$$

where the extra input v indexes the desired mapping $m_v(x,y)$.

Proof Sketch. Omitted due to lack of space. The proof follows a similar procedure to that used for Theorem 4.

The implication of this theorem is that by appropriate biases offsets in the different groups of units, various units in the recurrent network can be "pushed" in and out of action. Note that this method can be used as a means of switching between different discrete models, or in a continuous sense, by setting r to an appropriately small value.

6 Examples

In this section, we give a number of examples, which indicate the idea of the multiple models discussed in the paper. In order to clarify the results, we use simple model structures.

6.1 General Examples

Here we consider some general examples to indicate some possible types of multiple models which may be synthesized.

1. Input Amplitude Dependent Model.

   This type of model is derived by using a characteristic function M_c given by

$$M_c(x) = \begin{cases} \theta_0 x & x^2 \leqslant r_0 \\ \theta_1 x & r_0 < x^2 \leqslant r_1 \\ \theta_2 x & x^2 > r_2 \end{cases} \tag{21}$$

where $\{r\}$ are scalar values. The model behaviour is dependent on the instantaneous value of the input $x(t)$. In particular, the model will switch parameter sets when the amplitude of $x^2(t)$ crosses certain thresholds. This model can be considered as a multiple bilinear or state-dependent model.

2. Frequency Dependent Model Selection.

   M_c may be a function of the frequency of the incoming signal. For example,

   $$M_c(x) = f_m\left(X(\omega)\right) \tag{22}$$

   where f_m is a function of $X(\omega) = FFT[x]$.

3. Sequential Model Selection

   Here, the model is based on some predetermined time-sequence and bears a close relationship with hybrid systems considered in robotics [5]. In this case, the input x is a sequence of symbolic binary values. $M_c(x)$ processes this symbolic binary input and upon recognition of a particular sequence, outputs a 1 and holds it for a specified period of time, otherwise the output is a zero.

6.2 A Recurrent Network Multiple Model

6.2.1 Network Architecture

Based on the multiple model framework presented in this paper, a recurrent network multiple model[3] can be proposed as follows.

$$y(t) = \frac{q^{n-m}\prod_{i=1}^{m_1}\left(q - g_{\beta ri}\left(u_{\beta ri}\right)\right)\prod_{j=1}^{m_2}\left(q - g_{\beta ci}\left(u_{\beta ci}\right)\right)\left(q - g^*_{\beta ci}\left(u_{\beta ci}\right)\right)}{\prod_{i=1}^{n_1}\left(q - g_{\alpha ri}\left(u_{\alpha ri}\right)\right)\prod_{j=1}^{n_2}\left(q - g_{\alpha ri}\left(u_{\alpha ri}\right)\right)\left(q - g^*_{\alpha ri}\left(u_{\alpha ri}\right)\right)}x(t) \tag{23}$$

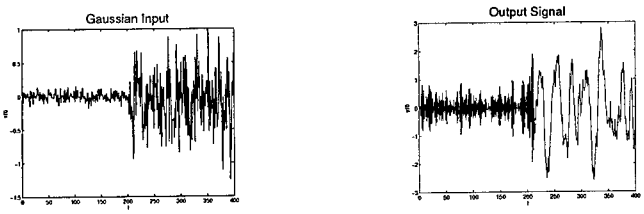
where $\{g\}$ are the element-wise characteristic functions corresponding to M_c. $\{u\}$ are ancillary inputs. Since the poles and zeros in (23) are the outputs of nonlinear functions, they are termed nonlinear poles and zeros respectively and the model can also be considered as a *nonlinear pole-zero model.* The coefficient functions $\{g\}$ and ancillary inputs$\{u\}$ provide a wide scope for introducing a variety of models.

6.2.2 Example: An Input Amplitude-Dependent Multiple Model

A nonlinear pole-zero multiple model is synthesized in this example as follows. The model is described by

$$\begin{aligned} H_m(q) &= \frac{1}{\left(1 - G_1(u)\,q^{-1}\right)\left(1 - G_1^*(u)q^{-1}\right)} \\ G_1(u) &= \alpha_1 u(t) + \alpha_2 \bar{u}(t) \qquad G_1^*(u) = \alpha_1^* u(t) + \alpha_2^* \bar{u}(t) \end{aligned} \tag{24}$$

[3] For convenience and clarity of the example, we have chosen to use linear models as a basis, however these may be arbitrary nonlinear models in practice. Since the behaviour of linear systems is well known, it is easier to visualize the differences between the two models in this case.



(a) (b)

Figure 1: The performance of a recurrent neural network multiple model. In this simple example, (a) is the input signal, and (b) shows the model output due to the input.

where $G_1(u)$ and $G_1^*(u)$ provide complex conjugate outputs and $H_m(q)$ switches between two underlying linear models $H_1(q)$, $H_2(q)$, depending on the binary ancillary input signal $u(t)$.

$$H_i(q) = \frac{1}{1 + a_{i1}q^{-1} + a_{i2}q^{-2}} \tag{25}$$

(26)

where, for the purposes of this example, we choose $a_{11} = 1.6$, $a_{12} = 0.73$, $a_{21} = -1.9$, $a_{22} = 0.925$ and α_i are the corresponding first order poles of $H_i(q)$. The input $u(t)$ can be obtained, for example, by

$$u(t) = \Gamma\left(E_s\left[x^2(t)\right]\right) \tag{27}$$

where $E_s[\cdot]$ denotes the short term expectation and

$$\Gamma(\varsigma) = \begin{cases} 1 & \varsigma \geqslant 0 \\ 0 & \varsigma < 0 \end{cases} \tag{28}$$

Therefore, when the magnitude of the short term average of the squared input signal reaches a certain threshold, defined by Γ, the model will change. The performance of this model is shown in Fig. 1, where the model characteristics due to the change in input can be seen.

6.2.3 Example: An RNN Modelling a Time-varying Linear System

This example we synthesize a recurrent neural network model which models a time-varying linear system, described by

$$H_m(q) = \frac{1}{1 - \zeta_1(u,t)\,q^{-1} + \zeta_2(u,t)\,q^{-2}} \tag{29}$$

The model varies as a function of the ancillary input signal u, and the coefficient functions $\zeta_i(t)$, $i = 1, 2$ are in this example, simple linear functions, given by

$$\zeta_i(t) = c_{i0} + c_{i1}\zeta_i'(t) \tag{30}$$

$$\zeta_i'(t) = \frac{b_{i0}u(t) + b_{i1}u(t-1)}{1 + a_{i1}q^{-1} + a_{i2}q^{-2}} \tag{31}$$

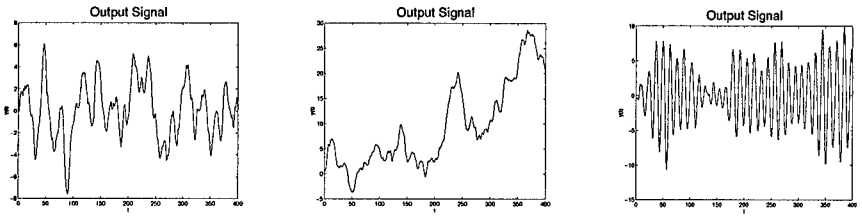


Figure 2: The time-varying recurrent network described in Section 6.2.3 is capable of exhibiting a variety of different behaviours as observed here.

For the purposes of this example, we choose the parameters[4] $a_{11} = 1.6$, $a_{12} = 0.9$, $a_{21} = -1.9$, $a_{22} = 0.925$, $b_{10} = 1.0$, $b_{11} = 0.5$, $b_{20} = 0.9$, $b_{21} = 0.5, c_{10} = 1.75$, $c_{11} = 0.04$, $c_{20} = 0.8$, $c_{21} = 0.02$ and $u(t) = x(t)$.

The resulting multiple model, which is an extension of the usual bilinear structure, can be described by the difference equation form of nonlinear pole-zero model given by

$$y(t) = x(t) + \zeta_1(t)\, y(t-1) + \zeta_2(t)\, y(t-2) \quad (32)$$

$$\zeta_i(t) = c_{i1} + c_{i2}\left(b_0 x(t) + b_1 x(t) - a_{i1}\zeta_i(t-1) + a_{i2}\zeta_i(t-2)\right) \quad (33)$$

where $x(t)$ is the input and $y(t)$ is the output. Examples of the model behaviour are shown in Fig. 2, which indicate some of the 'richness' of the model's capabilities.

7 Conclusions

In this paper, we have given theories which indicate how nonlinear models, including feedforward and recurrent networks, can approximate systems known as multiple models. We have shown that such models can be considered in terms of operational maps. It was shown that there exist classes of neural networks which can universally approximate operational maps. The results provide an explanation for the experimental behaviour of some recurrent networks in being able to model multiple dynamic systems [8].

Acknowledgments

The author acknowledges the support of the Frontier Research Program RIKEN and helpful discussions with S. Amari, T. Chen and L. Feldkamp.

[4]The parameters were chosen on the basis of known, stable linear systems. In the case of this linear model, it is necessary to bound the pole positions as usual. However, the situation changes when normal recurrent networks with sigmoid functions are employeed. The signals are then bounded by the sigmoids.

References

[1] A.D. Back and A.C. Tsoi. A cascade neural network model with nonlinear poles and zeros. In *Proc of 1996 Int Conf on Neural Information Processing ICONIP'96*, volume 1, pages 486–491, Piscataway, NJ, 1996. IEEE.

[2] C.J. Bett and M.D. Lemmon. Bounded amplitude control using multiple linear agents, technical report ISIS-97-004. Technical report, ISIS Group, University of Notre Dame, March 1997.

[3] S.P. Boyd. *Volterra Series: Engineering Fundamentals.* PhD thesis, University of California, Berkeley, 1985.

[4] M.S. Branicky. *Studies in Hybrid Systems.* PhD thesis, MIT, 1986.

[5] R. Brockett. Hybrid models for motion control systems. In *Perspectives in Control*, pages 29–54, Boston: Birkhauser, 1993.

[6] T. Chen and H. Chen. Approximations of continuous functionals by neural networks with application to dynamic systems. *IEEE Trans. Neural Networks*, 4(6):910–918, 1993.

[7] T. Chen and H. Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Trans. Neural Networks*, 6(4):911–917, 1995.

[8] L.A. Feldkamp, 1996. Personal communication.

[9] K. Funahashi and Y. Nakamura. Approximation of dynamical systems by continuous time recurrent neural networks. *Neural Networks*, 6:801–806, 1993.

[10] M.I. Jordan and R.A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6(2):181–214, 1994.

[11] D. Lainiotis. Partitioning: A unifying framework for adaptive systems I. estimation. *Proc. IEEE*, 64:1126–1143, 1976.

[12] K. Narendra, J. Balakrishnan, and M. Ciliz. Adaptation and learning using multiple models switching and tuning. *IEEE Control Systems Magazine*, 15(3):37–51, 1995.

[13] E. Ronco and P.J. Gawthrop. Modular neural networks: a state of the art. Technical Report CSC-95026, Center for System and Control, University of Glasgow, May 1995.

[14] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach,*. Prentice-Hall, Englewood Cliffs, NJ, 1995.

[15] I.W. Sandberg. Approximation theorems for discrete-time systems. *IEEE Trans. Circuits, Syst.*, 38(5):564–566, 1991.

[16] M. Schetzen. *The Volterra and Wiener-Theory of Nonlinear Systems.* Wiley, (Reprint Ed. Krieger Publishing, 1989), New York, NY, 1980.

[17] J. Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 1992.

[18] H.T. Siegelmann, B.G. Horne, and C.L. Giles. Computational capabilities of recurrent narx neural networks. *IEEE Trans. on Systems, Man and Cybernetics*, 26(6), 1996. In press.

[19] E. Sontag. Neural nets as systems models and controllers. In *Proc. Seventh Yale Workshop on Adaptive and Learning Systems*, pages 73–79. Yale University, 1992.

[20] R.J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1:270–280, 1989.

Induced Specialization of Context Units for Temporal Pattern Recognition and Reproduction

Ravi Kothari, and Kwabena Agyepong
Artificial Neural Systems Laboratory
Department of Electrical & Computer Engineering & Computer Science
University of Cincinnati
Cincinnati, OH 45221-0030
E-Mail: ravi.kothari@uc.edu

Abstract

Additional inputs to a feed-forward network, derived from the output of the hidden layer neurons, allow a feed-forward network to deal with temporal pattern recognition and reproduction tasks. These 'network derived' or 'context' inputs augment the 'true' inputs to the network and allow the network to retain past information necessary for temporal sequence processing. The choice of which hidden neurons to retain to provide the context inputs is difficult. Use of all the hidden neurons increases the size of the overall network resulting in poorer generalization performance. The problem is complicated due to difficulty in choosing the number of hidden layer neurons in the first place. In this paper, we propose the use of regularization terms in the sum-of-squared error cost function. Assuming the hidden layer neurons are indexed $1, 2, \ldots, m$, the regularization terms force the differentiation of hidden neurons 1 through m_1, and m_2 through m (where $1 < m_1 \leq m_2 < m$). Both m_1 and m_2 are controllable and allow fringe neurons to be used to provide the context inputs if the number of context units to use is known. When the number of context neurons to use cannot be determined, the regularization terms minimize m_1, and maximize m_2, while hidden neurons m_1 through m_2 are penalized for differentiation. An amplitude detection simulation is used to evaluate the efficacy of the proposed paradigm.

0-7803-4256-9/97/$10.00

I INTRODUCTION

Neural network models for temporal sequence processing can be broadly classified into [1],[2]:

1. *Tapped delay line models*: The network has *past inputs* explicitly available (through a tapped delay line) to determine its response at a given point in time (see for example [3]). Thus the temporal pattern is converted to a spatial pattern which can then be learned through, say, classic back-propagation [5].
2. *Context models or Partial recurrent models*: These models retain the *past output* of neurons instead of retaining the past raw inputs. For example, the output of the hidden layer neurons of a feed-forward network can be used as inputs to the network along with the true inputs [6]. These 'network derived' inputs are also called context inputs. When the interconnections carrying the context inputs are fixed, classic back-propagation can be used for training the network. More complex variations of this basic idea include self-feedback in the context inputs or deriving the context inputs from other locations in the network [3],[7].
3. *Fully recurrent models*: These models employ full feedback and interconnections between all units [8]-[10].

While tapped delay line models are the simplest to use and have been applied to such tasks as speech recognition (see for example [11]), one has to use a tapped delay line of length equal to the longest possible sequence to accommodate all the sequences. This increases the input dimensionality, and consequently the size of the network, requiring much more training data [12]. Alternatively, a larger network gives poorer generalization performance as compared to a smaller sized network when both networks are trained on the same amount of data. Fully recurrent models, which lie on the other end of the spectrum, are the most flexible but suffer from large time and storage requirements [1]. Context models lie somewhere between the simplicity of a tapped delay line model and the power of a fully recurrent network. For many sequence processing tasks they provide competitive solutions. It has also been shown that such context models can approximate the behavior of a finite state automaton [13].

In the actual implementation of context models, one is faced with the difficulty of selecting the hidden layer neurons which will provide the context inputs. One can obviously use the entire hidden layer but this results in an increase of the dimensionality of the augmented input space, thereby requiring additional training data. The problem is further complicated since the number of hidden layer neurons are not known in the first place. This paper is directed towards obtaining better generalization performance despite these difficulties.

Our approach is based on trying to localize the hidden neurons which differentiate while forcing the non-differentiating hidden layer neurons to have minimal activation. The localization of the hidden layer neurons is forced to appear at the fringes i.e. assuming the hidden layer neurons are indexed $1, 2, \ldots, m$, the regularization terms force the differentiation of hidden neu-

rons 1 through m_1, and m_2 through m (where $1 < m_1 \leq m_2 < m$). Both m_1 and m_2 are controllable and allow fringe neurons to be used to provide the context inputs if the number of context units to use is known. When the number of context neurons to use cannot be determined, the regularization terms minimize m_1, and maximize m_2, while hidden neurons m_1 through m_2 are penalized for differentiation. We attempt to enure this behavior through the use of regularization terms in the familiar sum-of-squared error cost function.

The rest of this paper is organized as follows. In the next section, we derive an algorithm that induces specialization of the fringe hidden layer neurons. In section 3, we illustrate the efficacy of the proposed model with simulations. In section 4, we present our conclusions.

II Induced Specialization of Context Units

We consider a standard feed-forward architecture with n inputs, a single hidden layer with m neurons, and an output layer with o neurons. The network operates synchronously and in a layered manner — i.e. all neurons in a layer are simultaneously updated, and then the next layer is updated and so on. The n inputs to the network consist of the 'true' inputs (x), and the context inputs which are the immediate past output of the fringe hidden layer neurons copied using one-to-one non-modifiable connections (see Figure 1). Since the contextual feedback is always derived from the fringe hidden layer neurons, our approach is to train the network using a cost function which includes, besides the sum-of-squared error term, regularization terms which force the fringe hidden layer neurons to differentiate.

We denote the network training data as consisting of p input-desired output pairs $\{(\xi^\mu, \zeta^\mu)\}$. ξ^μ is thus the augmented input vector comprising of the true inputs and the context inputs. The weight from input k to hidden layer neuron j is denoted by w_{jk} and the weight from hidden neuron j to output neuron i is denoted by W_{ij}.

The output of a hidden layer neuron (z_j^μ), on input pattern μ, is a non-linear function of its net input (h_j^μ) i.e,

$$z_j^\mu = f\left(h_j^\mu\right) = f\left(\sum_{k=1}^{n} w_{jk}\xi_k^\mu\right) \tag{1}$$

Similarly, the output of an output layer neuron (y_i^μ), on input pattern μ, is a non-linear (or linear) function of its net input (s_i^μ) i.e,

$$y_i^\mu = f\left(s_i^\mu\right) = f\left(\sum_{j=1}^{m} W_{ij}z_j^\mu\right) \tag{2}$$

To induce specialization of the context units we introduce regularization terms in the sum-of-squared error formulation of the cost function. Thus

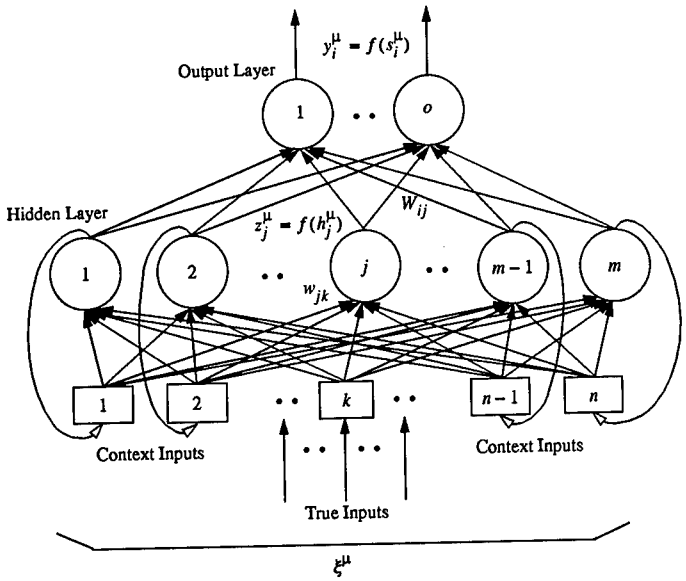


Figure 1: The architecture of the proposed network. The context inputs are derived from the fringe hidden layer neurons which are made to specialize through a regularization term added to the traditional sum-of-squared error cost function. Weights with hollow arrow-heads are fixed (at 1) and serve only to produce a copy of the activation from source to destination.

we define a cost function as:

$$J = \frac{1}{2}\sum_{\mu=1}^{p}\left[\sum_{i=1}^{o}(\zeta_i^\mu - y_i^\mu)^2 \;+\; \lambda_1\sum_{j=1}^{m} e^{-\frac{1}{2\sigma^2}(j-\frac{m}{2})^2} z_j^{\mu 2} \;+\; \lambda_2 e^{(\sigma-\theta)^2}\right] \quad (3)$$

where, λ_1 and λ_2 are Lagrange multipliers, and θ is explained shortly. Regularization is introduced in (3) through the use of two terms which we explain below.

The first of these terms consists of two components. The first component, $e^{-\frac{1}{2\sigma^2}(j-\frac{m}{2})^2}$, is a Gaussian which achieves its maximum at $m/2$ i.e. achieves a maximum for hidden layer neurons in the middle and falls of, at a rate governed by σ, as we approach the fringe hidden layer neurons. The second component of this first term, $z_j^{\mu 2}$ is the square of the activation of a hidden layer neuron j. The product of the two components ensures that neurons activations in the middle of the hidden layer (i.e. neurons with indices near $m/2$) are penalized. Consequently the fringe hidden layer neurons are made to specialize through this term allowing for contextual feedback to be drawn from them.

The second regularization term, $e^{(\sigma-\theta)^2}$, tries to make σ approach θ. θ should be chosen such that the Gaussian (in the first regularization term) has de-

cayed to a small value as one approaches the fringe hidden layer neurons selected for providing the contextual inputs. Consequently, the value of θ chosen reflects *how many* contextual inputs are required. Once the number of contextual inputs have been decided, one can simply choose them to be the fringe neurons. Alternatively, one can use all the hidden layer neurons to provide the context inputs while using $\theta = m/2$. This choice allows σ to be adapted to be the largest possible, or conversely, as few hidden layer neurons as possible to differentiate. The remaining hidden layer neurons which are fed back as context inputs do not differentiate (i.e. behave similarly) and consequently does not impede generalization performance.

Thus, the cost function in (3) forces the minimization of the SSE under the constraint that: (i) the chosen fringe hidden layer neurons differentiate the most, or (ii) when all the hidden layer neurons provide the context inputs, then as many of them of them behave similarly as possible.

We now proceed to derive the update equations. Since the context inputs are copied from the output of the hidden layer neurons through non-modifiable weights, the update equations are easily obtained from (3) by performing gradient descent in the weight space, and adjusting the weights proportional to the negative of the gradient. We thus obtain for the output to hidden weights:

$$\begin{aligned}\Delta W_{ij} &= -\eta \frac{\partial J}{\partial W_{ij}} \\ &= \eta \sum_{\mu=1}^{p} \left[(\zeta_i^\mu - y_i^\mu) f'(s_i^\mu) z_j^\mu \right] = \eta \sum_{\mu=1}^{p} \left[\delta_i^\mu z_j^\mu \right] \end{aligned} \tag{4}$$

where, $\delta_i^\mu = (\zeta_i^\mu - y_i^\mu) f'(s_i^\mu)$, and η is a constant of proportionality referred to as the learning rate.

Similarly, the weight update equations for the hidden to input weights are:

$$\begin{aligned}\Delta w_{jk} &= -\eta \frac{\partial J}{\partial w_{jk}} \\ &= \eta \sum_{\mu=1}^{p} \left[\left(\sum_{i=1}^{o} \delta_i^\mu W_{ij} \right) f'(h_j^\mu) \xi_k^\mu \; - \right. \\ &\qquad \left. \lambda_1 e^{-\frac{1}{2\sigma^2}(j-\frac{m}{2})^2} z_j^\mu \left(f'(h_j^\mu) \xi_k^\mu \right) \right] \end{aligned} \tag{5}$$

Finally, the update equation for σ is:

$$\begin{aligned}\Delta \sigma &= -\eta \frac{\partial J}{\partial \sigma} \\ &= -\eta \sum_{\mu=1}^{p} \left[\frac{\lambda_1}{\sigma^3} \sum_{j=1}^{m} e^{-\frac{1}{2\sigma^2}(j-\frac{m}{2})^2} z_j^{\mu 2} (j - \frac{m}{2})^2 \; + \right. \end{aligned}$$

$$\lambda_2 e^{(\sigma-\theta)^2}(\sigma-\theta)\Big] \tag{6}$$

where constants have been absorbed into the Lagrange multipliers. The training consists of choosing the number of context inputs. If the choice can be made, then the context inputs should be derived from the fringes. If the choice cannot be made, then one can use $\theta = m/2$ and use all the hidden neurons to provide the context inputs. Patterns are then continually presented in sequence, and parameters are updated (in batch or pattern mode) according to equations (4)–(6) until the error is within the desired tolerance. To prevent large numbers it is best to scale σ and θ such that they are less than or equal to 1. In the next section we present a simulation to illustrate the efficacy of the proposed approach.

III Simulation

The simulation we present deals with amplitude detection in a signal formed by concatenating sinusoids of fixed frequency but varying amplitude[1] The entire data set is shown in Figure 1. A sine wave of four different amplitudes (1.0, 2.0, 1.6, and 1.2) are used to generate the data with 20 points sampled from each amplitude. We use the first 40 points (corresponding to an amplitude of 1 and 2) for training and reserve the remaining 40 points (corresponding to amplitudes of 1.6 and 1.2) for generalization. Observe that it is not possible to estimate the amplitude of the underlying sine wave at a given point in time without examining more than one successive sample.

We considered a network of 1 (true) input, 15 sigmoidal hidden neurons, and 1 output. We considered two distinct situations. In the first situation, we used a total of 4 context inputs (2 drawn from the left and 2 drawn from the right fringe). Consequently, θ was selected so that the Gaussian in equation (3) would decay to a near zero value at the 2^{nd} neuron from the left and right. This simulation thus simulates the condition where the number of context neurons to use has been determined. In the second situation we used all the hidden neurons to provide the context inputs simulating the condition where the number of context neurons to use are unknown. We refer to a context network trained with back-propagation as a Context Network (CN) and a context network trained with the proposed Induced Specialization as (CNIS). For all cases, we trained the network using the first 40 points to the same sum-of-squared error (0.5). Figure 3 shows the results when 4 hidden layer neurons (2 each from the left and right fringe) are used for providing the context inputs while Figure 4 shows the results when all the hidden layer neurons are used to provide the context inputs. It is clear that in both cases CNIS performs better. Table I to quantifies the differences.

[1]This problem is given as a demonstration of a standard Elman network (called CN in this paper) in the neural network toolbox of Matlab. Though the results in this paper are generated from our own code, we were inspired to use it since the network has to decide on an amplitude that it was never trained with. It thus serves as a good test of generalization.

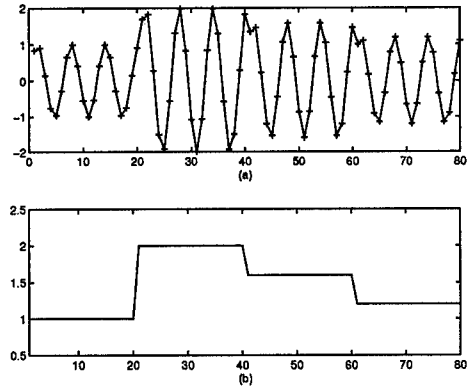


Figure 2: The complete data used for the simulations. (a) shows the input while (b) shows the desired output. The desired output of the network at a point in time is is the amplitude sine wave at that time. The first 40 points are used for training. The remaining 40 points are used for obtaining the generalization performance.

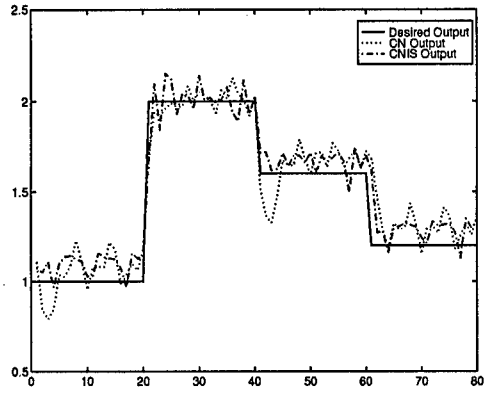


Figure 3: Performance comparison of the standard context network, and the proposed context network with induced specialization. A network of 1 true input, 15 sigmoidal hidden layer neurons, and 1 linear output is used. For the standard context network (CN) and the context network with induced specialization (CNIS), 4 hidden layer neurons (2 each from the left and right fringe) are used to provide the context input. Both networks are trained to the same error (0.5) on the first 40 points.

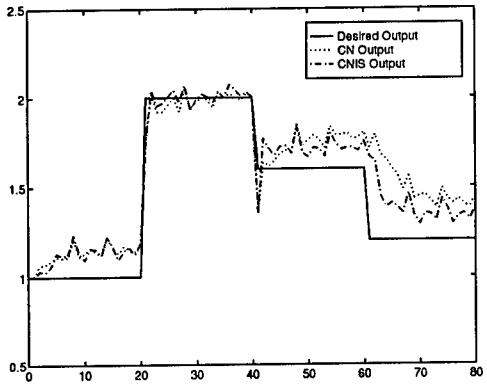


Figure 4: Performance comparison of the standard context network, and the proposed context network with induced specialization. A network of 1 true input, 15 sigmoidal hidden layer neurons, and 1 linear output is used. For the standard context network (CN) and the context network with induced specialization (CNIS), all 15 hidden layer neurons provide context input. Both networks are trained to the same error (0.5) on the first 40 points.

# Context Inputs	CN	CNIS
4	0.9394	0.7227
15	1.6466	1.1512

Table: I: Sum-of-squared error on generalization. The network used in all cases had 1 'true' input, 15 sigmoidal hidden layer neurons, and 1 linear output neuron. All networks were trained to the same error (0.5) on the first 40 points.

IV Conclusion

We proposed the use of regularization terms in the standard sum-of-squared error function to allow for fringe hidden layer neurons to differentiate, while penalizing the differentiation of the neurons in the middle of the hidden layer. This induced specialization allowed contextual feedback to be always drawn from the fringe hidden layer neurons allowing for a network with better generalization properties. Initial results indicate improved generalization performance with the proposed paradigm. While approaches such as pruning (see for example [14]) can be used to obtain better generalization there are two points in favor of the proposed paradigm:

- Pruning approaches typically estimate the sensitivity of the error to a weight and discard the weight if the output error does not depend on the particular weight. Consequently, composite effects of weight removals are not considered. Since the entire network is trained in the proposed paradigm, composite effects are included.
- One often requires retraining after pruning a weight (though methods to distribute the role of the weight to be removed have been proposed).

Of course, there appears to be no reason why pruning approaches could not be applied after training with the proposed paradigm.

References

[1] J. Hertz, and R. G. Palmer, *Introduction to the Theory of Neural Computation*, Massachusetts, Addison-Wesley, 1991.

[2] M. C. Mozer, "Neural Net Architectures for Temporal Sequence Processing," In *Time Series Prediction: Forecasting the Future and Understanding the Past*, A. S. Weigend, and N. A. Gershenfeld (eds.), pp. 243-264, MA, Addision-Wesley, 1993.

[3] M. C. Mozer, "A Focused Back-Propagation Algorithm for Temporal Pattern Recognition," *Complex Systems*, Vol. 3, pp. 349-381, 1989.

[4] B. de Vries, and J. C. Principe, "The Gamma Model — A New Neural Network Model for Temporal Processing," *Neural Networks*, Vol. 5, pp. 565-576, 1992.

[5] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Back-Propagating Errors," *Nature*, vol. 332, pp. 533-536, 1986.

[6] J. L. Elman, "Finding Structure in Time," *Cognitive Science*, Vol. 14, pp. 179-211, 1990.

[7] M. I. Jordan, "Serial Order: A Parallel Distributed Processing Approach," In *Advances in Connectionist Theory: Speech*, J. L. Elman, and D. E. Rumelhart (eds.), 1989.

[8] R. J. Williams, and D. Zipser, "A Learning Algorithm for Continually Running Fully Recurrent Networks," *Neural Computation*, Vol. 1, pp. 270-280, 1989.

[9] M. Sato, "A Real Time Learning Algorithm for Recurrent Analog Neural Networks," *Biological Cybernetics*, Vol. 62, pp. 237-241, 1990.

[10] F. J. Pineda, "Generalization of Back-Propagation to Recurrent Neural Networks," *Physical Review Letters*, Vol. 59, pp. 2229-2232, 1987.

[11] A. Waibel, "Modular Construction of Time-Delay Neural Networks for Speech Recognition," *Neural Computation*, Vol. 1, pp. 39-46, 1989.

[12] E. B. Baum, and D. Haussler, "What Sized Net Gives Valid Generalization," *Neural Computation*, Vol. 1, pp. 151-160, 1989.

[13] A. Cleermans, D. Servan-Schreiber, and J. L. McClelland, "Finite State Automata and Simple Recurrent Networks," *Neural Computation*, Vol. 1, pp. 372-381, 1989.

[14] B. Hassibi, and D. G. Stork, "Second order derivatives for networks pruning: optimal brain surgeon," *Proc. Neural Information Processing Systems 4*, pp. 164-171, 1993.

UNIFORM APPROXIMATION AND THE COMPLEXITY OF NEURAL NETWORKS

Paulo J. S. G. Ferreira*, Si-Qi Cao

Departamento de Electrónica e Telecomunicações
Universidade de Aveiro
3810 Aveiro Portugal

E-mail: pjf@inesca.pt

Abstract

This work studies some of the approximating properties of feedforward neural networks as a function of the number of nodes. Two cases are considered: sigmoidal and radial basis function networks. Bounds for the approximation error are given. The methods through which we arrive at the bounds are constructive. The error studied is the L_∞ or sup error.

1 STATEMENT OF THE PROBLEM

Let $x \in \mathbb{R}^n$. Feed-forward sigmoidal neural networks compute

$$\sum_{i=1}^{N} c_i \, F(g_i(x)), \tag{1}$$

where $F : \mathbb{R} \to \mathbb{R}$ is sigmoidal and

$$g_i(x) = \sum_{j=1}^{n} g_{ij} x_j - a_i.$$

The approximation properties of these neural networks as a function of N are of theoretical and practical interest.

Another interesting type of network is the so-called radial basis function neural network, in which case (1) is replaced with

$$\sum_{i=1}^{N} c_i \, G(h_i(x)),$$

where

$$h_i(x) = \|x - x_i\|_2 \quad (x_i \in \mathbb{R}^n).$$

In this case, the set of possible choices for G includes the Gaussian function

$$\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}.$$

*This work was partially upported by JNICT.

0-7803-4256-9/97/$10.00 ©1997 IEEE

but also

$$\sqrt{|x|^2+c}, \quad \frac{1}{\sqrt{|x|^2+c}}, \quad \frac{x}{c^2}\log\frac{x}{c}.$$

The purpose of this paper is to study the following nonlinear approximation problems.

Problem 1 *Consider the function*

$$a(x) = \sum_{i=1}^{N} c_i\, F(g_i(x)),$$

where F is sigmoidal and $g_i : \mathbb{R}^n \to \mathbb{R}$ is

$$g_i(x) = \sum_{j=1}^{n} g_{ij} x_j - a_i.$$

The function clearly depends on a number of parameters (the coefficients c_i, g_{ij} and a_i). Given the function $f : D \subset \mathbb{R}^n \to \mathbb{R}$, the problem is to select N and then the coefficients in such a way that the L_∞ error

$$e = \sup_{x\in D} |f(x) - a(x)|$$

is small. We wish to study the dependence of e upon N.

Problem 2 *Consider the function*

$$a(x) = \sum_{i=1}^{N} c_i\, G(h_i(x)),$$

where G is Gaussian and $h_i : \mathbb{R}^n \to \mathbb{R}$ is

$$h_i(x) = \|x - x_i\|_2.$$

The function clearly depends on a number of parameters (the coefficients c_i, the parameters of the Gaussian G, the vectors x_i). Given the function $f : D \subset \mathbb{R}^n \to \mathbb{R}$, the problem is to select N and then the parameters in such a way that the L_∞ error

$$e = \sup_{x\in D} |f(x) - a(x)|$$

is small. We wish to study the dependence of e upon N.

The equivalent one-dimensional problems, although considerably simpler, are also of interest (primarily because they suggest methods that might work for the general case).

Problem 3 *Given $f : [0,1] \to \mathbb{R}$, study the L_∞ error associated with the approximation*

$$f(x) \approx \sum_{i=1}^{N} c_i\, F(g_i x - a_i),$$

where F is sigmoidal.

Problem 4 *Given $f : [0,1] \to \mathbb{R}$, study the L_∞ error associated with the approximation*

$$f(x) \approx \sum_{i=1}^{N} c_i G(x - x_i)$$

where G is the Gaussian

$$\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}.$$

2 RELATED WORK

A recent overview of neural networks and their relevance to signal processing applications can be found in [7]. For a study of radial basis neural networks see [9].

The approximation properties of superpositions of sigmoidal functions were studied by Cybenko [2]. Park and Sandberg [10] studied the approximation properties of radial basis function neural networks. Barron [1] gave striking bounds for the approximation error committed when approximating a given function $f : \mathbb{R}^n \to \mathbb{R}$ by superpositions of a sigmoidal function. The bounds, surprisingly, do not depend on the dimension n of the space considered.

Barron considers the mean square error (the error in the norm of L_2), whereas we consider uniform approximation and consequently the sup norm (or the L_∞ norm). Mean square and uniform convergence are distinct concepts: a sequence of functions may converge in the norm of L_2 but not in the sup norm. Two given functions f and g may differ by very little in the L_2 norm, while differing by arbitrarily large numbers in the L_∞ norm. On the other hand, if the domain of the functions is a compact interval, convergence in L_∞ does imply mean square convergence. Thus, the L_∞ error, despite the challenging analytical difficulties that it often poses, may help in understanding these approximation problems.

The approximation problem discussed here was studied in [6] for the Gaussian, one-dimensional case. The approach that we use, in contrast to others, is constructive (it yields the values of the parameters, as well as the bounds). It also relates to some recently obtained results concerning nonuniform sampling approximations [3–5].

Recently, we came across [11], which also addresses the uniform approximation problem[1] and presents an interesting treatment of the approximating power of a network. The bounds given are distinct from ours. Asymptotically, they are weaker (a brief comparison is outlined below).

3 THE ONE-DIMENSIONAL PROBLEMS

The notation $f \in BV$ means that the function f is of bounded variation. The value of the variation itself is $V(f)$. The support of a function f is denoted by $supp(f)$.

[1] I am grateful to Prof. Bock (Aachen, Germany) for bringing this work to my attention and kindly supplying a copy.

3.1 Gaussian-based radial basis functions

It follows readily from the work of Norbert Wiener [12] on the closure of translations that superpositions of Gaussians are dense in the L_1 and L_2 spaces.

Take L_1, for example, and let $\psi \in L_1$. Wiener showed that any L_1 function can be approximated in the L_1 norm by

$$\sum_{i=1}^{N} c_i \psi(t - t_i),$$

iff the Fourier transform $\hat{\psi}$ has no zeros. The Fourier transform of the Gaussian certainly has no zeros, a fact that immediately implies the universal approximation properties of sets of translated Gaussians.

However, Wiener's proof is not constructive. We will show, with the help of a constructive procedure, how to pick c_i and t_i and to obtain a bound for the L_∞ approximation error.

The idea is to consider the convolution

$$f_\sigma(t) = \int_{-\infty}^{+\infty} f(\tau) G(t - \tau, \sigma)\, d\tau,$$

where $G(x, \sigma)$ is Gaussian with standard deviation σ, f is the function to approximate, and f_σ denotes the result of the convolution (which depends upon σ through the convolution kernel). Next, we approximate f by f_σ, and f_σ by a finite sum. Since the approximation of f by f_σ is well-known it remains to study the error that arises when approximating the convolution by the finite sum. This is the purpose of the following theorem, in which the support of f is assumed to be the interval $[0, 1]$.

Theorem 1 *Let $f : \mathbb{R} \to \mathbb{R}$ be BV, with supp$(f) \subset [0, 1]$. Denote by $\{t_i\}$ any N reals such that*

$$\frac{i-1}{N} < t_i < \frac{i}{N}, \tag{2}$$

for $1 \leq i \leq N$, and let

$$f_\sigma(t) = \int_0^1 f(\tau) G(t - \tau, \sigma)\, d\tau.$$

Then,

$$\left| f_\sigma(t) - \frac{1}{N} \sum_{k=1}^{N} f(t_k) G(t - t_k, \sigma) \right| \leq \frac{1}{\sigma N} \frac{V(f) + \|f\|_\infty}{\sqrt{2\pi}}. \tag{3}$$

Proof: By the mean-value theorem, there exists in the intervals defined by (2) points $\{\xi_i\}_{1 \leq i \leq N}$ such that

$$f_\sigma(t) = \frac{1}{N} \sum_{i=1}^{N} f(\xi_i) G(t - \xi_i, \sigma).$$

This shows that

$$\left| f_\sigma(t) - \frac{1}{N} \sum_{k=1}^{N} f(t_k) G(t - t_k, \sigma) \right| \leq V[f(x) G(t - x, \sigma)].$$

The result follows after evaluating the variation of the product. ∎

As σ grows, $f_\sigma \to f$ in the pointwise sense, provided that f has some regularity. For any fixed σ, it always possible to select N such that (3) becomes less than any specified positive number. Hence, it is always possible to obtain arbitrarily good approximations to f:

$$\|f - s_N\| \leq \|f - f_\sigma\| + \|f_\sigma - s_N\|$$

in any norm.

We now turn to approximation by superposition of sigmoidal functions.

3.2 SIGMOIDAL FUNCTIONS

Definition 1 *A bounded function $F : \mathbb{R} \to \mathbb{R}$ is sigmoidal if*

$$\lim_{x \to +\infty} F(x) = 1, \quad \lim_{x \to -\infty} F(x) = 0.$$

In the following, the dilation $\psi(wx)$ of any function ψ is denoted by $\psi_w(x)$, and the function u is the unit step function. It is assumed without loss of generality that the functions f to approximate satisfy $f(0) = 0$.

Lemma 1 *If F is sigmoidal, $f \in BV$, and $supp(f) \subset [0,1]$,*

$$\left| f(t) - \int_0^1 f'(x) F_w(t-x)\, dx \right| \leq \epsilon V(f),$$

uniformly in t, for all ω such that $|u(x) - F_w(x)| \leq \epsilon$.

Proof: Assume that $f(0) = 0$. The fact that $f \in BV$ justifies the identity

$$\begin{aligned} f(t) &= \int_0^t f'(t-x) u(x)\, dx \\ &= \int_0^1 f'(x) u(t-x)\, dx, \end{aligned}$$

and so

$$f(t) - \int_0^1 f'(x) F_w(t-x)\, dx = \int_0^t f'(t-x)[u(x) - F_w(x)]\, dx.$$

If $x \neq 0$, given any $\epsilon > 0$ there is a $\Omega > 0$ such that

$$|u(x) - F_w(x)| < \epsilon$$

if $\omega > \Omega$. Thus,

$$\left| f(t) - \int_0^1 f'(x) F_w(t-x)\, dx \right| \leq \epsilon \int_0^t |f'(t-x)|\, dx \leq \epsilon V(f).$$

∎

Lemma 2 *Take any N reals $\{t_k\}_{1\le k\le N}$ satisfying*

$$\frac{k-1}{N} \le t_k \le \frac{k}{N}, \quad k = 1, 2, \ldots, N,$$

and consider the functions

$$\begin{aligned} f_w(t) &= \int_0^1 f'(x) F_w(t-x)\, dx \\ &= \int_0^t f'(t-x) F_w(x)\, dx, \end{aligned}$$

$$s_N(t) = \frac{1}{N} \sum_{k=1}^{N} f'(t_k) F_w(t - t_k).$$

Then

$$|f_w(t) - s_N(t)| \le \frac{V(f')\|F_w\|_\infty + \|f'\|_\infty V(F_w)}{N},$$

assuming $f' \in BV$, $\phi \in BV$.

Proof: Similar to the proof of theorem 1: use the mean-value theorem, then bound the variation $V[f'(x)F_w(t-x)]$. ∎

Theorem 2 *For any $\epsilon > 0$ there is w such that*

$$|f(t) - s_N(t)| = O\left(\frac{V(f')}{N}\right).$$

for any $\{t_k\}_{1\le k\le N}$ satisfying

$$\frac{k-1}{N} \le t_k \le \frac{k}{N}, \quad k = 1, 2, \ldots, N.$$

Proof: We have

$$\|f - s_N\| \le \|f - f_w\| + \|f_w - s_N\|.$$

The first term converges to zero as $w \to \infty$, and the second term is bounded by a term inversely proportional to N (lemma 2). The first term can be made smaller than $\epsilon/2$ by picking w sufficiently large. Once w is fixed, take N so large that $\|f_w - s_N\| \le \epsilon/2$. ∎

It is not necessary to have f' of bounded variation. The restriction may be removed by approximating it by an absolutely continuous function h' such that

$$|f'(t) - h'(t)| \le \epsilon, \quad \int_0^1 |f'(t) - h'(t)|\, dt \le \eta.$$

4 THE MULTIDIMENSIONAL PROBLEM

In this section we sketch the solution to the multidimensional problem for radial basis neural networks. The statement and proofs of the results depend on certain number theoretic results concerning uniform distribution, discrepancy, and numerical integration [8]. We start with the following inequality

$$\left| \int_I F(x)dx - \frac{1}{N} \sum_{i=1}^{N} F(x_i) \right| \leq D_N V(F),$$

where D_N denotes the discrepancy of the sequence $x_1, x_2, \ldots x_N$ and $V(F)$ is the total variation of the function $F \in BV$ (which is assumed to be of bounded variation in the sense of Hardy and Krause [8]). Note that $x \in \mathbb{R}^n$, as well as each x_i. Standard results in the estimation of the discrepancy D_N show that

$$D_N \leq \frac{\log^{n-1} N}{N}$$

if the sequence x_i is a good lattice set. It is possible to show that the variation in the sense of Hardy and Krause of a product of two functions $V(fg)$ is bounded by an expression of the form

$$V(fg) = \|f\|_\infty V(g) + \|g\|_\infty V(f) + O(\|f\|_\infty \|g\|_\infty).$$

Letting $F(x) = f(x)g(t-x, \sigma)$ leads to a bound of the form

$$s_N(x) = \frac{1}{N} \sum_{i=1}^{N} f(x_i) g(x - x_i, \sigma),$$

$$|f_\sigma(x) - s_N(x)| = O\left(\frac{\log^{n-1} N}{N} \right).$$

The constant hidden by the O notation depends on σ, but not on N. The total error satisfies

$$\|f - s_N\| \leq \|f - f_\sigma\| + \|f_\sigma - s_N\|.$$

The first term is a function of σ but not of N, and can be studied by well known methods (the precise bound depends on the regularity of f). The second term is $O(\log^{n-1} N/N)$, and predicts a degradation of performance as the dimension n of the space increases. The bound given in the interesting work of Ritter [11] is $O(1/\sqrt[n]{N})$ (for sigmoidal networks). This is is weaker for sufficiently large N, but better for smaller N.

References

[1] A. R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39(3):930–945, May 1993.

[2] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2:303–314, 1989.

[3] P. J. S. G. Ferreira. Nonuniform sampling of nonbandlimited signals. *IEEE Signal Processing Letters*, 2(5):89–91, May 1995.

[4] P. J. S. G. Ferreira. Approximation by nonuniform sampling series and multiresolution analysis. In *Proceedings of the 7th IEEE Digital Signal Processing Workshop, DSPW-96*, pages 137–140, Loen, Norway, September 1996.

[5] P. J. S. G. Ferreira. On the approximation of nonbandlimited signals by nonuniform sampling series. In *Proceedings of EUSIPCO-96, VIII European Signal Processing Conference*, pages 1567–1570, Trieste, Italy, September 1996.

[6] P. J. S. G. Ferreira. Neural networks and approximation by superposition of Gaussians. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 97*, volume IV, pages 3197–3200, Munich, Germany, April 1997.

[7] S. Haykin. Neural networks expand SP's horizons. *IEEE Signal Processing Magazine*, 13(2):24–49, March 1996.

[8] L. Kuipers and H. Niederreiter. *Uniform Distribution of Sequences.* John Wiley & Sons, New York, 1974.

[9] B. Mulgrew. Applying radial basis functions. *IEEE Signal Processing Magazine*, 13(2):50–65, March 1996.

[10] J. Park and I. W. Sandberg. Universal approximation using radial-basis-function networks. *Neural Computation*, 3:245–257, 1991.

[11] G. Ritter. Jackson's theorems and the number of hidden units in neural networks for uniform approximation. Technical Report MIP-9415, Universität Passau, Fakultät für Mathematik und Informatik, D-94030 Passau, Germany, December 1994.

[12] N. Wiener. *The Fourier Integral and Certain of Its Applications.* Cambridge University Press, Cambridge, 1933. Reprinted by Dover Publications, 1958.

BIOMEDICAL

Invited Lecture

Neural Networks for Medical Image Processing

David G. Brown, PhD

Division of Electronics and Computer Science
Center for Devices and Radiological Health
Rockville, MD

The past two decades have witnessed an explosion of medical imaging technologies. The invention of computed tomography served as a vital bridge between our analog, plain-film past and our increasingly digital present, endowed with a multiplicity of sophisticated new imaging modalities. These new systems are exemplified by the complex world of magnetic resonance imaging, but include many other types of systems as well. In addition to use as components for the functioning of these imaging systems, neural networks are increasingly being used for image processing tasks such as segmentation and identification of anatomic or functional structures. Computer-aided diagnosis (CADx) is becoming increasingly important as data becomes available in digital form--and becoming available in overwhelming quantities now, frequently from three and four (time) dimensional imaging data sets. Neural networks are being used in commercially available as well as research CADx systems. In addition, they are being applied to such problems as image registration for multiple single-modality or multiple modality images. Neural networks have demonstrated a growing importance in this dynamic field of modern medicine.

Mixture of Discriminative Learning Experts of Constant Sensitivity for Automated Cytology Screening

Jenq-Neng Hwang, Eugene Lin

Information Processing Laboratory
Dept. of Electrical Engineering, Box # 352500
University of Washington, Seattle, WA 98195

Abstract

One practical objective in an automated cytology screening task is to obtain as high as possible specificity (the percentage of normal slides being classified as normal) while attaining acceptable (predefined) constant sensitivity. In this paper, we propose a new learning algorithm which continuously improves the specificity while maintaining constant sensitivity for pattern classification problems. We further propose to integrate the pre-trained networks with constant sensitivities into the mixture of experts (MOE) network configuration. This enables each trained expert to be responsive to specific subregions of the input spaces with minimum ambiguity and thus produces better performance.

1 Introduction

Thanks to recent advances in image processing technologies and classification algorithms, the automated cytology screening system has gained commercial interest [5]. In an automated cytology screener, the object classification module classifies the Pap smeared object as a "normal" or an "abnormal" slide. Sensitivity, which is defined as the percentage of abnormal slides being correctly classified as abnormal, is a very important factor to most automated biomedical applications. One practical objective in training a neural network for these applications is to obtain as high as possible specificity (the percentage of

0-7803-4256-9/97/$10.00

normal slides being classified as normal) while attaining acceptable (predefined) sensitivity.

The traditional mechanism to achieve this objective is to train the classifiers first and then generate the Receiver Operating Characteristics (ROC) curve based on different thresholds. By fixing a constant sensitivity, the corresponding threshold and specificity can thus be determined. If the specificity is not acceptable, then the whole training process, which might involve a new classifier structure, needs to be reinitiated until the acceptable specificity is attained under the predefined sensitivity level. In reality, the inherent tradeoff between sensitivity and specificity prevents high specificity in case of high sensitivity. Therefore, we would like to explore a neural network learning procedure that can overcome this problem directly in the learning phase without the need of varying the thresholds after training or reinitiating the learning. In this paper, we adopt the discriminative learning neural network techniques [4, 3] for automated cytology screening and propose a new learning procedure which can obtain high specificity while maintaining an acceptable constant sensitivity.

The discriminative learning of a feedforward network distinguishes itself from the traditional backpropagation learning by having a different cost function. The presence of the discriminative cost function has a profound impact on the learning capability and performance of the network, and usually results in better performance. Built upon the discriminative learning algorithm, we propose the constant sensitivity learning procedure which continuously improves the specificity while maintaining constant sensitivity for pattern classification problems.

Mixture of experts (MOE) learning [2] has been shown to provide better performance due to its ability to effectively solve a large complicated task by smaller and modularized trainable networks (i.e., experts), whose solutions are dynamically integrated into a coherent one using the trainable gating network. One of the major concern in using the MOE is the lack of a meaningful interpretation of each trained expert when all the expert networks and the gating network are trained simultaneously. This concern is further amplified when using the MOE for medical diagnosis applications, where each reasoning step leading toward the final decision needs to be self-explanatory. It is highly desired to have each trained expert to be responsive to specific (non-overlapping) subregions of the input space so that the gating network can unambiguously identify and integrate the correct solution. In this paper, we thus propose to pre-train each expert net-

work which operates in a fixed sensitivity accomplished by the modified discriminative learning strategy. This enables the MOE network to systematically identify a good operating point in the Receiver Operating Characteristics (ROC) curve and thus produces better performance. This approach is based on the principle of divide-and-conquer such that each expert network can represent some distinct subregions of the input space with the minimum amount of overlap so that the gating network's output probabilities associated with those subregions can be quite distinct, i.e., a clear winner can be easily identified.

We applied the proposed constant sensitivity procedure to automated cytology screening tasks. The proposed discriminative training with constant sensitivity outperforms the traditional backpropagation learning and the discriminative learning without enforcing the constant sensitivity. Furthermore, the proposed mixture of discriminative learning experts of constant sensitivity (MDLECS) architecture further improves the performance and outperforms the standard MOE techniques.

2 Discriminative Learning

The discriminative learning [4, 3] was proposed specifically for pattern recognition problems, aiming at achieving a minimum classification error rate. Based on a given set of training samples, the objective criterion is defined by the classification rule in a functional form and is optimized by numerical search algorithms. Under the backpropagation learning framework, the discriminant functions, $\{f_i(\mathbf{x};\mathbf{W}), i = 1, 2, ..., M\}$, which are the neural network outputs and indicate the classification posterior probabilities $P(i|\mathbf{x})$ [6], are first calculated, where $\mathbf{W}$ denotes the parameter set of the classifier (i.e., the feedforward network weights $\{w_{ij}(l)\}$ in the l-th layer) and the training sample $\mathbf{x}$ is known to belong to one of M classes. For each input $\mathbf{x}$, the classifier makes its decision by choosing the largest of the discriminants evaluated on $\mathbf{x}$. A misclassification measure for this data is then defined as follows:

$$d_i(\mathbf{x}) = -f_i(\mathbf{x};\mathbf{W}) + \left[\frac{1}{M-1}\sum_{j,j\neq i}^{M} f_i(\mathbf{x};\mathbf{W})^{\eta}\right]^{\frac{1}{\eta}}, \qquad (1)$$

where η is a positive number.

Finally, the minimum error objective is formulated and is expressed as a differentiable function of the misclassification measure. More specifically, the error objective function E_k of the k-th class is defined as

$$E_k(\mathbf{x};\mathbf{W}) = E_k(d_k(\mathbf{x})) = \frac{1}{1+e^{-\tau d_k}}, \quad \tau > 0. \tag{2}$$

Note that a positive $d_k(\mathbf{x})$ leads to a penalty which is a count of classification error, while a negative $d_k(\mathbf{x})$ implies a correct classification.

3 Constant Sensitivity Discriminative Learning

To achieve high specificity while maintaining constant sensitivity, we propose a new procedure built upon the discriminative learning algorithm on a feedforward neural network [7], i.e., a 2-layer perceptron. The training sample x is known to belong to one of 2 classes: the abnormal slides as Class 1 and the normal slides as Class 2. We also define a constant λ_0 which is the sensitivity value to be maintained during the optimization process. In this 2-class application, with η approaches ∞ in Eq. (1), it results in the simple misclassification measure $d_k(\mathbf{x})$:

$$d_1(\mathbf{x}) = -f_1 + f_2, \qquad d_2(\mathbf{x}) = -f_2 + f_1. \tag{3}$$

Instead of using the (sigmoid) error objective function E_k of Eq. (2), we also tried another error objective function:

$$E_k(\mathbf{x};\mathbf{W}) = E_k(d_k(\mathbf{x})) = c_1 * d_k^{\tau} + c_2 * d_k^{\tau-1}, \tag{4}$$

where $c_1 > 0$, $c_2 > 0$, and $\tau > 0$. In our simulation, $c_1 = 1$, $c_2 = 2$, and $\tau = 2$ are used, that is,

$$E_k(\mathbf{x};\mathbf{W}) = E_k(d_k(\mathbf{x})) = d_k^2 + 2 * d_k. \tag{5}$$

The optimization process, which increases the specificity over all the normal training data while maintaining constant sensitivity λ_0 over all the abnormal training data, uses the simple iterative gradient descent search algorithm by separately dealing with E_1 and E_2. A "batch" training procedure is used, i.e., the weights won't be updated until the accumulation of weight changes of all training samples in one iteration. The calculation of sensitivity is always based on the threshold of 0.5.

1. For each training input $\{\mathrm{x}^{(n)}\}$ in Class 1 (abnormal):

$$w_{ij}(l) \Longleftarrow w_{ij}(l) + \alpha_1 \sum_n \frac{\partial E_1}{\partial w_{ij}(l)}, \quad if\ \lambda_{abnormal} \geq \lambda_0 \ (6)$$

$$w_{ij}(l) \Longleftarrow w_{ij}(l) - \alpha_1 \sum_n \frac{\partial E_1}{\partial w_{ij}(l)}, \quad if\ \lambda_{abnormal} < \lambda_0 \ (7)$$

   where $\lambda_{abnormal}$ denotes the sensitivity value evaluated over all abnormal training samples at the previous iteration.

2. For each training input $\{\mathrm{x}^{(n)}\}$ in Class 2 (normal):

$$w_{ij}(l) \Longleftarrow w_{ij}(l) - \alpha_2 \sum_n \frac{\partial E_2}{\partial w_{ij}(l)}. \quad (8)$$

In case of oscillation of $\lambda_{abnormal}$ around λ_0, the updating steps, α_1 and α_2, are gradually decreased.

4 Mixture of Experts

To achieve the goal of a meaningful interpretation of experts in a mixture of experts (MOE) network and also to have each trained expert to be responsive to non-overlapping subregions of the input spaces, we further integrate the proposed constant sensitivity learning algorithm into a mixture of experts architecture [2], where each expert network is pre-trained with different constant sensitivity.

For a given input x, the total probability of generating class y from x based on a K-expert MOE is computed by:

$$P(\mathrm{y}|\mathrm{x}, \phi) = \sum_{i=1}^{K} g_i P(\mathrm{y}|\mathrm{x}, \theta_i) \quad (9)$$

where y is a binary vector, e.g., y is either $[1,0]$ or $[0,1]$ for a 2-class problem. ϕ is the set of parameters associated with the gating and the expert networks. $\{g_i\}$ are the gating probabilities for weighting the expert outputs, $\{P(\mathrm{y}|\mathrm{x}, \{\theta_i\})\}$, and $\{\theta_i\}$ are the parameters for the i-th expert network $(i = 1, ..., K)$.

The gating network can be a nonlinear neural network or a linear combiner. To obtain the gating network probabilistic outputs, the softmax function is adopted [1]. The learning algorithm for the MOE

is based on the maximum likelihood principle to estimate the parameters, the synaptic weights associated with the gating and the expert networks. In this training procedure, the resulting threshold is always set as 0.5. The gradient ascent algorithm can be used to estimate the parameters. Since the expert networks have been pre-trained by the constant sensitivity discriminative learning algorithm, therefore the parameters for the expert networks remain unchanged during the training processing of the MOE. Note that, in this mixture of discriminative learning experts of constant sensitivity (MDLECS) configuration, the overall sensitivity value can not be prefixed (it is within the range of our minimum and maximum sensitivities provided by the expert networks). An obvious advantage of this architecture is that the training won't be over-generalized, which is always the case when much fewer abnormal training data are used and the trained network normally get pushed to very high specificity with very low sensitivity when standard network training algorithms are used.

5 Comparative Simulation Results

To verify the feasibility and superiority of the proposed constant sensitivity discriminative learning procedure and its integration with MOE, we utilized the real world cytology data provided by NeoPath Inc., who developed and manufactured the world's first automated Pap Smear screening system - the AutoPap 300. There are 32 features extracted from the Pap Smear images obtained from a single slide. All these features were normalized to have zero mean and unit standard deviation before input to the classifiers. Among these data, 5000 slide samples (2880 "abnormal", 2120 "normal") are used as the training set, and another 5000 slide samples (2880 "abnormal", 2120 "normal") are used as the independent testing set.

The overall accuracy, i.e., classification rate over the testing data, is used to evaluate the performance of classifiers. Table 1 shows the overall accuracy of the 5000 testing data under different sensitivity values i.e., λ is equal to 0.75, 0.8, 0.85, 0.9, and 0.95. Three learning procedures are carefully experimented and compared: the backpropagation learning, the discriminative learning, and the constant sensitivity discriminative learning. It appears that the performance of the constant sensitivity discriminative learning outperforms the other two learning methods. Also note that for the constant sensitivity discriminative

learning, the use of the error objective function in Eq. (4) provides better accuracy than that in Eq. (2). All these simulations use one-hidden layer feedforward neural network with the same size of hidden units.

Based on the five networks pre-trained by the constant sensitivity discriminative learning, we further built the mixture of experts with a linear gating network. Each expert network is a one-hidden layer feedforward neural network with 10 hidden units, and the gating network is a single-layered feedforward neural network with the softmax outputs. The overall accuracy of the mixture of experts is 78.54 % with an overall sensitivity of 0.82. The performance was further improved to 79.04 % (with a sensitivity of 0.81) when a nonlinear gating network was used (i.e., 2-layer feedforward perceptron with 20 hidden units). These performance are favorably compared with that of a 2-layer feedforward perceptron trained at constant sensitivity (fixed at 0.80) discriminative learning with 50 hidden units (77.56% with testing sensitivity 0.77), which has similar size of parameters as the proposed MDLECS (we also tried the 70-unit 2-layer perceptron, the performance is almost the same as 50-unit one). Note specifically a higher sensitivity is consistently achieved by the proposed mixture of discriminative learning experts of constant sensitivity (MDLECS). To have a fair comparison, we also trained a standard MOE network without pre-trained constant sensitivity experts. Each of the five expert networks has 10 hidden neurons and the nonlinear gating network contains 20 hidden neurons, both expert and gating networks are trained by gradient ascent algorithm simultaneously. The resulting accuracy is 78.18 % (with a sensitivity of 0.79).

By observing the gating network output probabilities associated with 5 experts, we are able to find some clues to explain the better performance of the proposed MDLECS. Table 3 shows 12 representative output probabilities of the gating network trained with the standard MOE procedure. Note that the MOE did a good job in separating the input space into subregions, therefore only one dominating probability exists in most cases. The proposed MDLECS did a even more clear-cut partition of input subregions, therefore the one-dominating probability situation is even more obvious for the same set of testing data, as evidenced by the bold face numbers in Table 3.

Sensitivity λ_0	0.75	0.8	0.85	0.9	0.95
Backpropagation	75.70	75.88	73.32	70.38	65.56
Discriminative	76.68	76.62	74.82	71.88	66.46
Constant Sensitivity with Eq. (2)	76.92	77.16	76.26	74.68	69.50
Constant Sensitivity with Eq. (4)	77.40	78.16	77.78	75.50	71.28

Table 1: The comparative testing accuracy (% correct) for the comparative simulations among three learning procedures.

Methods	Testing Accuracy	Sensitivity
Fixed Constant Sensitivity BP (32-50-2)	77.56%	0.80
Standard MOE	78.18%	0.79
MDLECS (Linear Gating)	78.54%	0.82
MDLECS (Nonlinear Gating)	79.04%	0.81

Table 2: The comparative testing accuracy (% correct) and the resulting sensitivity for various mixture of experts configurations.

MOE	g_1	g_2	g_3	g_4	g_5
Data 1	0.23463	0.00089	**0.75585**	0.00420	0.00441
Data 2	**0.56412**	0.06864	0.35065	0.01592	0.00064
Data 3	**0.91152**	0.01505	0.06893	0.00423	0.00023
Data 4	0.06379	0.01670	0.04422	**0.84404**	0.03123
Data 5	**0.64777**	0.02383	0.08398	0.20182	0.04258
Data 6	**0.78372**	0.00229	0.20898	0.00474	0.00024
Data 7	**0.56004**	0.00370	0.42796	0.00713	0.00115
Data 8	0.05941	0.07737	0.15883	**0.63616**	0.06820
Data 9	0.41742	0.00432	**0.57676**	0.00120	0.00027
Data 10	**0.53426**	0.00380	0.46015	0.00164	0.00012
Data 11	0.15195	0.30548	0.05661	**0.48406**	0.00188
Data 12	**0.87367**	0.00080	0.12464	0.00082	0.00004
MDLECS	g_1	g_2	g_3	g_4	g_5
Data 1	**0.94406**	0.00002	0.00017	0.00008	0.05564
Data 2	0.02298	0.00167	0.04568	0.00610	**0.92354**
Data 3	0.01396	0.00158	0.04408	0.00779	**0.93257**
Data 4	0.37157	**0.61927**	0.00115	0.00529	0.00269
Data 5	**0.96350**	0.02500	0.00029	0.00840	0.00278
Data 6	**0.72072**	0.00299	0.00796	0.00046	0.26785
Data 7	**0.88323**	0.00381	0.00403	0.00120	0.10770
Data 8	0.05907	0.00004	0.00001	**0.94084**	0.00002
Data 9	**0.99580**	0.00332	0.00007	0.00008	0.00071
Data 10	**0.95901**	0.04066	0.00001	0.00000	0.00030
Data 11	**0.77697**	0.04275	0.15508	0.00580	0.01938
Data 12	**0.86628**	0.00741	0.00025	0.00000	0.12604

Table 3: The output probabilities of the gating network for 12 representative testing data trained with the standard MOE and MDLECS procedures.

6 Conclusion

In this paper, we present a new learning procedure built upon the discriminative learning algorithm to achieve high accuracy while maintaining constant sensitivity. We further integrate these trained constant sensitivity networks into the mixture of experts (MOE) configuration, which results in very encouraging simulation performance in automated cytology screening applications.

References

[1] J. S. Bridle, "Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition," In F. Fogelman-Soulie and J. Hérault, editors, Neuro-computing: Algorithms, Architectures and Applications, pp. 227-236, Springer-Verlag, 1991.

[2] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive Mixtures of Local Experts," Neural Computation, 3:79-87, 1991.

[3] B. H. Juang and S. Katagiri, "Discriminative Learning for Minimum Error Classification," IEEE Trans. on Signal Processing, vol. 40, no. 12, pp. 3043-3054, 1992.

[4] S. Katagiri, C. H. Lee, and B. H. Juang, "Discriminative Multilayer Feedforward Networks," 1991 IEEE Workshop Neural Networks for Signal Processing, pp. 11-20, Princeton NJ, 1991.

[5] J. S. J. Lee, J. N. Hwang, D. T. Davis, and A. C. Nelson, "Integration of Neural Networks and Decision Tree Classifiers For Automated Cytology Screening," in Proc. Int'l Joint Conf. on Neural Networks, I:257-262, Seattle WA, July 1991.

[6] M. D. Richard and R. P. Lippmann, "Neural Network Classifiers Estimate Bayesian A Posterior Probabilities," Neural Computation, 3(4):461-483, 1991.

[7] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning internal representation by error propagation," in Parallel Distributed processing: Explorations in the Micro- structure of Cognition: Volume 1: Foundations, MIT Press, Cambridge, MA, 1986.

DYNAMICS MODELLING IN BRAIN CIRCULATION

R. Silipo, C. Schittenkopf, G. Deco, A. Brawanski †
Siemens AG, Dept ZT IK4, D-81730 Munich (Germany)
†Clinic of Neurosurgery, University of Regensburg, D-93042 Regensburg (Germany)

Abstract

Two different measurement modalities, one related to blood flow, the other related to brain metabolism are monitored in a head injury patient and analyzed by using the method of surrogate data. That is applied against a hierarchy of two-dimensional Markov processes, designed to model a possible deterministic behaviour of the system and correlations between the two observed variables. Two-layered feed-forward neural networks are trained to estimate the two-dimensional conditional densities of the proposed Markov models. A cumulant based information flow is here used for testing the observed dynamics against the hierarchy of null hypotheses. A deterministic dynamics corresponding to a low order Markov process was found in both time series. In addition some correlation was detected indicating a coupling of the blood flow and the metabolism related parameters depending on patient condition. The proposed method could be an useful tool for detecting malfunction in the regulation of the human basic metabolism and predicting its evolution inside a reasonable window time.

1. MONITORING OXYGEN METABOLISM IN THE BRAIN

The measurement of the brain metabolism and related parameters is becoming a more and more helpful tool to assess the follow up of people with serious trauma of the central nervous system.

In a head injury, first the space occupying lesions is removed surgically, whenever necessary. Then the brain edema is treated not surgically by hyperventilation, sedation, and osmotic agents. Specifically the conservative treatment is guided by various parameters which are recorded on line in the Intensive Care Unit and give hints of the metabolic state of the brain. However treatment guided by these parameters is reactive. That means that whenever the parameters give an indication of a deterioration, treatment is adjusted. It is highly desirable to know those episodes of deterioration in ad-

0-7803-4256-9/97/$10.00

vance, since the treatment could be more appropriate and avoid crisis. Quite often very similar changes in the parameter trends can lead to very different consequences. It would be useful to evaluate whether the time course of such variables in head injured patients shows any deterministic structure and then whether it is possible to characterize those clinical situations.

In this work we analyze time series of data coming from two devices during different coma conditions of the same patient. One parameter is the local partial oxygen pressure of frontal white matter of the brain (tipO2). This parameter is more related to blood flow and diffusion dependent oxygen delivery. The second parameter defines the oxygen loading of hemoglobin in venous and arterial blood, and is more related to metabolism (HgBO2).

We want to analyse two issues in this study. First of all the Markovian character of the underlying process is investigated, in order to model the system structure and to perform some prediction on the future values. Then a possible correlation of the two signals is studied. A strong correlation would indicate an intact regulation of the brain metabolism, whereas no correlation would indicate a serious disruption of this regulatory mechanism.

A statistical approach for detecting the Markovian character of dynamical systems by analyzing their information flow is here applied to the two signals (tiPO2 and HgBO2) alone or in combination, in order to detect deterministic behaviour and/or correlation.

A measure based on higher order cumulants depending on the past values of both time series is calculated. That quantifies the statistical dependences of a point r steps ahead in one of the two time series on their past values. These cumulant based information flows, expressed as a function of the lookahead r, are here used as a discriminating statistics in testing the observed dynamics against a hierarchy of null hypotheses, corresponding to two-dimensional nonlinear Markov processes of increasing order [1]. The process can be reiterated generating higher and higher order Markov models, corresponding to better approximations of the underlying two-dimensional process in terms of information flow.

Two-layered feed-forward neural networks are trained to estimate the conditional densities of each Markov process.

Not too many studies have been performed in this area yet, since the technological development only recently made available sophisticated measurement systems. Previous studies were oriented to analyze the case under a strictly medical point of view [6] or to check for chaotic behaviour [5].

Some deterministic dynamics, corresponding to a low order Markov process, was detected in both time series. A strong correlation was also detected in normal conditions of the patient, while one-dimensional Markov models were sufficient to describe both information flows when the patient was in worse clinical conditions.

2. TESTING NONLINEAR MARKOVIAN BIVARIATE HYPOTHESIS

Four cumulant based measures of the statistical dependences of the points r steps ahead in the time series x_t and y_t are derived based on n_x and n_y succeeding observations of the two time series. They indirectly describe the loss of information in the underlying dynamical system. Such measures are here used as a discriminating statistics to accept or reject a null hypothesis consisting of a Markov model supposed to be adequate to explain the system.

Let us define the $n_x + n_y + 1$-dimensional vector

$$\mathbf{v}_d = (x_{t-n_x}, \ldots, x_{t-1}, y_{t-n_y}, \ldots, y_{t-1}, d_{t+r})^T \tag{1}$$

where d_{t+r} represents alternatively x_{t+r} and y_{t+r}, then the general equation of those measures can be defined as:

$$m_{d\{d\}}(r) = \sum_{j=1}^{\infty} \sum_{i_1 \ldots i_j = L_d}^{U_d} K_{d i_1, \ldots, i_j, n_x+n_y+1} \tag{2}$$

where the $K_{d i_1 \ldots i_j}$ represent the cumulant of order j calculated on the vector $\mathbf{v}_d$ and d refers to the time series x_t or y_t. The dependency of x_{t+r} from the past n_x values of x_t is measured by $m_{xx}(r)$, calculated by using the vector $\mathbf{v}_{d=x}$ with $L_d = 1$, and $U_d = n_x$. Extending the calculation to the following n_y values in the vector $\mathbf{v}_x$ a measure $m_x(r)$ of the dependency of x_{t+r} itself from the past of both time series is obtained. In this case $U_d = n_x + n_y$. Similar measures can be defined for the time series y_t. A measure $m_{yy}(r)$ of the dependency of y_{t+r} on its past is produced by using $d_{t+r} = y_{t+r}$ in vector $\mathbf{v}_{d=y}$ with $L_d = n_x$ and $U_d = n_x + n_y$ in the eq. 2. Extending the sum to the previous n_x values in the vector $\mathbf{v}_y$, that means setting $L_d = 1$, a measure $m_y(r)$ of dependency of y_{t+r} on the past of both time series is obtained.

These four measures, $m_x(r), m_y(r), m_{xx}(r)$, and $m_{yy}(r)$, are a cumulant based characterization of the information flow of the two-dimensional underlying system and measure the dependences of x_{t+r} and y_{t+r} respectively from their n_x and n_y past values or from both. They can also be seen as measures of the dependence of x_t on y_t and viceversa. The first sum in 2 is approximated with a finite number of terms, and the cumulants are calculated up to the fourth order [1].

The method of surrogate data [2] is separately applied to the two time series. We define a null hypothesis as a two-dimensional Markov model supposed to be an adequate explanation of the underlying system. A discriminating statistics involving the cumulant based measures, $m_x(r)$, $m_y(r)$, $m_{xx}(r)$, and $m_{yy}(r)$, allows us to quantify the consistency of our null hypothesis with the property of the original system. The two-sample Student t test [1] is adopted as a discriminating statistics, expressed by the following variable

(3), which has a t student distribution with $M-1$ freedom degrees, M being the number of used surrogate data sets:

$$t_d(r) = \sqrt{\frac{M}{M+1}}\,\frac{\mu_{\hat{d}}(r) - m_d(r)}{\sigma_{\hat{d}}(r)}; \tag{3}$$

$$\mu_{\hat{d}}(r) = \frac{1}{M}\sum_{i=1}^{M} m_{\hat{d}_i}(r); \qquad \sigma_{\hat{d}}(r) = \sqrt{\frac{1}{M-1}\sum_{i=1}^{M}(m_{\hat{d}_i}(r) - \mu_{\hat{d}}(r))^2}; \tag{4}$$

where d represents the process x_t or y_t, $\hat{d}_i$ indicates the corresponding i^{th} surrogate time series, and r the lookahead. The null hypothesis is accepted if the absolute value $|t(r)|$ is smaller than $\hat{t}$ corresponding to a p-value, which depends on M [1], for lookahed $1 \leq r \leq 6$ for all the four cumulant based measures. In this case our assumption about the original time series is adequate. A hierarchy of null hypothesis is defined increasing the order of the Markov model, whenever the null hypothesis is rejected.

Before beginning the analysis, the time series data are independently gaussianized to avoid possible static nonlinearities due to the measurement process [2]. For that purpose, Gaussian random numbers are computed and re-ordered so that the ranks of both sequences – the original and the gaussianized one – agree. The next step is to determine the order $\{n_x, n_y\}$ of the two-dimensional nonlinear Markov process which is supposed to approximate the information flow of the transformed sequences. If we do not have detailed knowledge about the observed dynamical system we start with $n_x = 1, n_y = 0$. We do not deal with the case $n_x = 0, n_y = 0$ (white noise in both signals, since physicians already find some information in those signals).

Two-layered feed-forward neural networks are trained to perform an estimation of the conditional densities $p\left(x_{t+r}|x_t, \ldots, x_{t-n_x}, y_t, \ldots, y_{t-n_y}\right)$ and $p\left(y_{t+r}|x_t, \ldots, x_{t-n_x}, y_t, \ldots, y_{t-n_y}\right)$ of the proposed Markov model.

In order to provide to the networks the minimum number of significant inputs, a time delay is applied to the original time series. The first minimum of the mutual information, calculated as in [7], is adopted as the best available systematic criterion for choosing time delays.

Since it has been shown that *nonlinear* neural networks are very suitable for this purpose [4] we decided to approximate all the density functions with:

$$p\left(x_{t+r}|x_t, \ldots, x_{t-n_x}, y_t, \ldots, y_{t-n_y}\right) = \sum_{h=1}^{k} \frac{u_h}{\sqrt{2\pi\sigma_h^2}} \exp\left(-\frac{(x_{t+r} - \mu_h)^2}{2\sigma_h^2}\right), \tag{5}$$

$$u_h = v_{h0} + \sum_{i=1}^{l} v_{hi} \tanh\left(w_{hi0} + \sum_{j=1}^{n_x} w_{hij} x_{t-j} + \sum_{z=1}^{n_y} w_{hi(n_x+z)} y_{t-z}\right), \tag{6}$$

$$\mu_h = \tilde{v}_{h0} + \sum_{i=1}^{l} \tilde{v}_{hi} \tanh\left(\tilde{w}_{hi0} + \sum_{j=1}^{n_x} \tilde{w}_{hij} x_{t-j} + \sum_{z=1}^{n_y} \tilde{w}_{hi(n_x+z)} y_{t-z}\right), \tag{7}$$

$$\sigma_h^2 = \hat{v}_{h0} + \sum_{i=1}^{l} \hat{v}_{hi} \tanh\left(\hat{w}_{hi0} + \sum_{j=1}^{n_x} \hat{w}_{hij} x_{t-j} + \sum_{z=1}^{n_y} \hat{w}_{hi(n_x+z)} y_{t-z}\right) \quad (8)$$

where k denotes the number of Gaussians, l is the number of hidden neurons and the $u_h, v_{hi}, w_{hij}, \tilde{v}_{hi}, \tilde{w}_{hij}, \hat{v}_{hi}$, and $\hat{w}_{hij}$ are the parameters of the networks. Additionally, the constraint $\sum_{h=1}^{k} u_h = 1$ holds to ensure that the sum (5) is really a density function.

Each conditional probability density is therefore represented by a weighted sum of normal distributions, the weights, means, and variances of which are the outputs of different neural networks (multi-layer perceptrons, (6) - (8)) for the $(n_x + n_y)$-dimensional input. The training is performed following the maximum likelihood principle [3] in the sense that the parameters of the networks are updated according to the gradient descent on the log-likelihood function:

$$-\sum_t \log p(d_{t+r}; u_1, \mu_1, \sigma_1^2, \ldots, u_k, \mu_k, \sigma_k^2). \quad (9)$$

After the training the networks can build new sequences using the Markovian approximation of the information flow of the time series x_t and y_t starting from the first n_x values of x_t and the first n_y values of y_t. The new sequences $\hat{x}_i$ and $\hat{y}_i$, (*a surrogate data set*), are gaussianized to have the same marginal distribution as the original ones. An arbitrary number M of surrogate data sets are generated and the measures $m_{\hat{x}}(r)$, $m_{\hat{y}}(r)$, $m_{\hat{x}\hat{x}}(r)$, and $m_{\hat{y}\hat{y}}(r)$, are independently calculated for each one. If the hypothesis that the nonlinear Markov process of order $\{n_x, n_y\}$ is appropriate to explain the data, is rejected, the order is increased to $n_x + 1$ or $n_y + 1$, and the procedure is repeated starting with the training of the neural networks. The hypothesis is finally accepted if both tests are accepted for lookaheads $1 \leq r \leq 6$.

Starting with $n_x = 1$ and $n_y = 0$ and gradually increasing the order of the Markov process makes the detection of the appropriate Markov model with minimum order possible. Correlations between the two time series are also detected, if the Markov model with minimum order requires both n_x and n_y different from 0. The complete method is resumed in figure 1.

3. RESULTS

Two time series of oxygen and metabolism related parameters of the brain are here analyzed in a comatous patient after head injury. They are recorded at two different times, corresponding to two different patient conditions. During time A a clinically relatively stationary condition was recorded, while during time B the patient was in critical conditions.

The first measure trend (tipO2) invasively measures the local tissue oxygen tension in the white matter of the brain. The second technique locally measures the oxygen loading of hemoglobine (HgBO2) in arterial and venous blood of the brain. Both techniques adopt a $4s/m$ sample frequency.

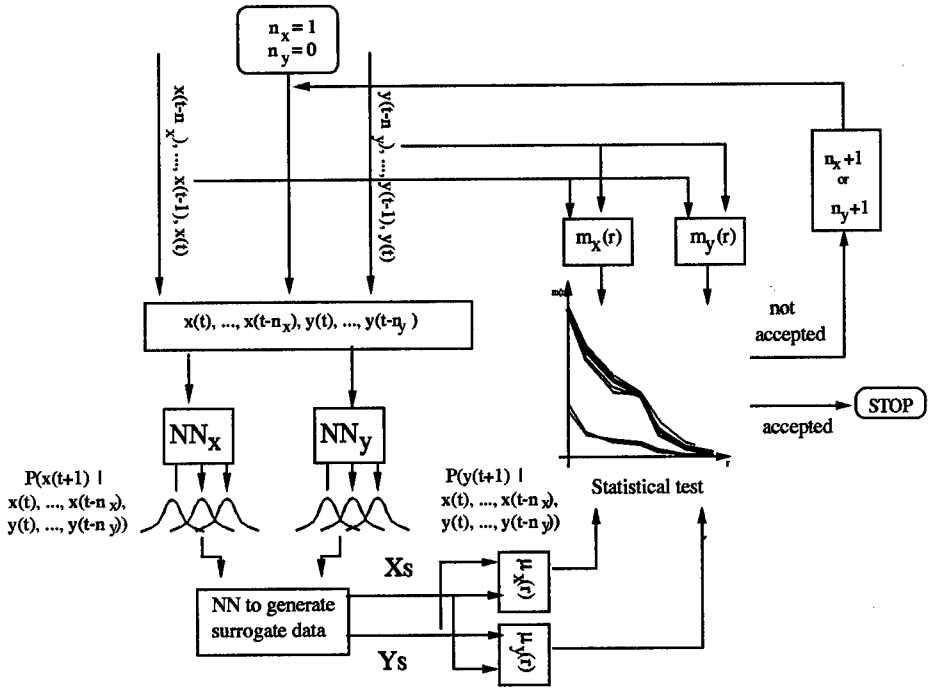


Figure 1: Block diagram for the minimum order model selection.

The method of surrogate data is applied to the two time series against a hierarchy of two-dimensional Markov processes, looking for correlations between the two signals and for deterministic behaviour.

In the figures 2 and 3, the cumulant based discriminating measures of the original data, – $m_x(r)$, $m_{xx}(r)$, $m_y(r)$, $m_{yy}(r)$ (2) – and the corresponding averaged ones of the surrogate data – $\mu_{\hat{x}\hat{x}}(r)$, $\mu_{\hat{x}}(r)$, $\mu_{\hat{y}\hat{y}}(r)$, $\mu_{\hat{y}}(r)$ (4) – for different Markov models are displayed as functions of the lookahead r. The order of the Markov models is indicated in parentheses. In both cases, it is easy to observe the progressive approaching of the averaged discriminating measures of the surrogate data to the original one with the increasing of the Markov order.

Even though the experiments are still in progress to extend the analysis to more patients, some deterministic dynamics, corresponding to a low order two-dimensional Markov process, was detected in both time series at both times A and B.

The order $\{1, 1\}$ is the lowest order of the Markov processes being not refused by the discriminating analysis of the two time series at time A (Fig. 2). It is easy to see in figure 2 that the statistical properties of the surrogate data, represented by the variables $\mu_{\hat{x}}(r)$, $\mu_{\hat{y}}(r)$, $\mu_{\hat{x}\hat{x}}(r)$, and $\mu_{\hat{y}\hat{y}}(r)$, fit the

ones of the original time series. In addition, while the tipO2 is well modelled by a one-dimensional Markov model of order $\{1, 0\}$ (Fig. 2.a), the minimum order of the Markov model needs some past values of the tipO2, in order to represent the HgBO2 too. That shows a strong correlation between the two time series, due to coupling of the two paramters indicating an intact regulation.

For the time series referring to the time B, two different one-dimensional Markov models are detected, respectively of order $\{2, 0\}$ for the tipO2 and $\{0, 1\}$ for the HgBO2 (Fig. 3). That describes two statistically independent processes. In this case coupling of blood flow and metabolism is disrupted indicating a serious disturbance of regulatory mechanisms.

We can see that the combined model of order $\{2, 1\}$ approximates the statistical behaviour of the two time series even better. That proves the efficiency of the proposed algorithm, since Markov processes with limited higher orders are still able to model the original process.

Both cases, A and B, show a system characterized by a low order dynamics, but the high amount of noise in the measurement process could hide higher order dynamics. We can not retrieve any information from the biological knowledge about the modelling of the oxygen metabolism in patients with serious head injury, since the problem involves too many variables and often the biological processes of interest are not completely known. In this sense, the method here proposed represents a good tool to investigate and to model the dynamics of the brain circulation, without a priori biological knowledge. Finally the proposed algorithm should allow to distinguish apparently similar conditions of the patient and possible pathologies in the brain metabolism. The knowledge in advance of the regulation of the basic metabolism of the patient would allow appropriate therapies, avoiding the ineffective ones.

4. CONCLUSIONS

The measurement of the oxygen and metabolism related parameters in the brain is a very helpful tool in assessing the follow up of patients with serious trauma of the central nervous system. Several variables related with oxygen delivery and metabolism in the brain can be monitored with different techniques. In addition, the forecasting and the modelling of such variables would be really important to undertake an appropriate therapy.

In this work, two different measurement trends related with oxygen metabolism in the brain are examined in two different clinical situations of the same patient.

The method of surrogate data is applied to the two time series against a hierarchy of two-dimensional Markovian processes, looking for correlation between the two signals and deterministic behaviour. It might be possible in that way to characterize similar clinical situations with different follow up by means of different dynamics.

Two-layered feed-forward neural networks are trained to perform the estimation of the two-dimensional conditional densities of the Markov models.

A deterministic dynamics, corresponding to a low order two-dimensional Markov process, was found in both time series for both clinical conditions. A correlation between the two time series was detected due to the control action of the autonomic nervous system on the human basic metabolism only when the patient was in better clinical conditions.

The proposed method could be an useful tool for modelling and detecting malfunctions in the regulation of the basic metabolism and to predict inside a reasonable time window its evolution. This in advance knowledge would allow appropriate therapies, avoiding the ineffective ones.

REFERENCES

[1] C. Schittenkopf, and G. Deco, "Testing nonlinear Markovian Hypotheses in Dynamical Systems.", **Physica D**, to appear.

[2] J. Theiler, S. Eubank, A. Longtin, B. Galdrikian, and J.D. Farmer, "Testing for nonlinearity in time series: the method of surrogate data.", **Physica D**, vol. 58, pp. 77-94, 1992.

[3] A. Papoulis, **Probability, Random Variables, and Statistic Processes**, Singapore, Mc Graw Hill, 1991.

[4] C. M. Bishop, **Neural Networks for Pattern Recognition** Oxford, Clarendon Press, 1995.

[5] A. Brawanski, M. Holzschuh, "Evidence for Low Dimensional Chaotic Dynamics in Brain Circulation.", working paper.

[6] W. Fleckenstein, A.I.R. Maas, G. Nollert, D.A. De Jong, **Blackwell Ueberreuter Wissenschaft**, Berlin, Eds. Ehrly *et al.* , pp. 368-395, 1990.

[7] A.M. Fraser, H.L. Swinney, "Independent coordinates for strange attractors from mutual information", **Physical Review A**, vol. 33, n. 2, pp. 1134-1140, 1986.

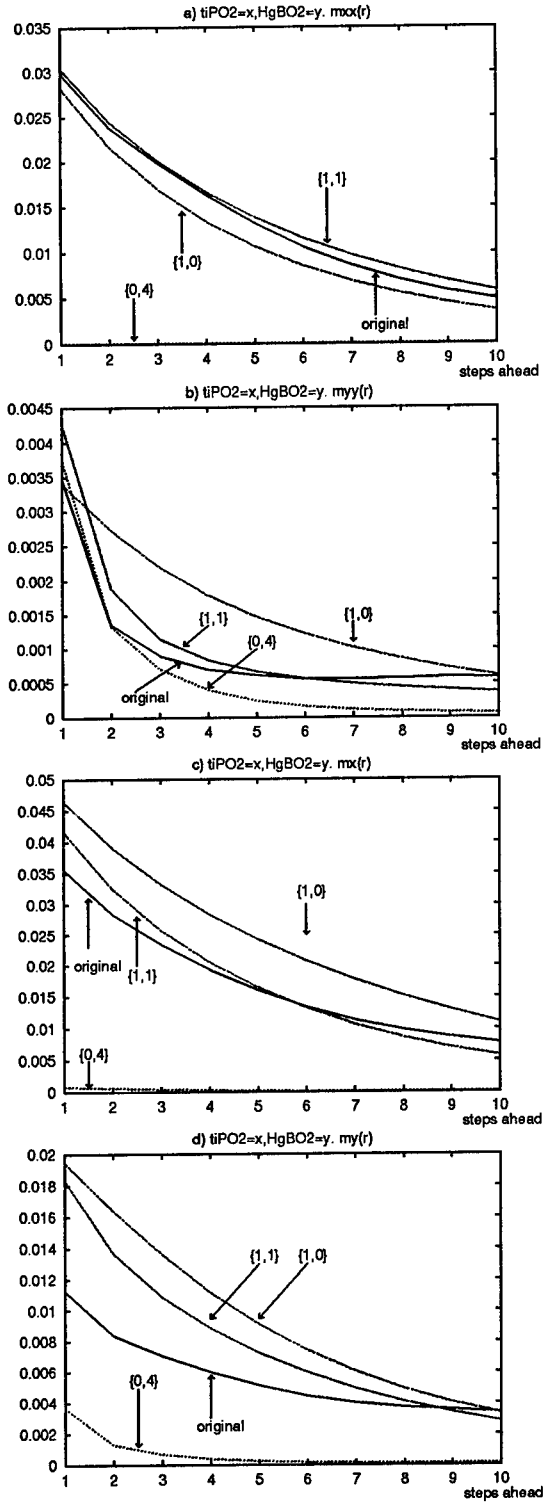


Figure 2: Modelling of HgBO2 and tiPO2 at the time A.

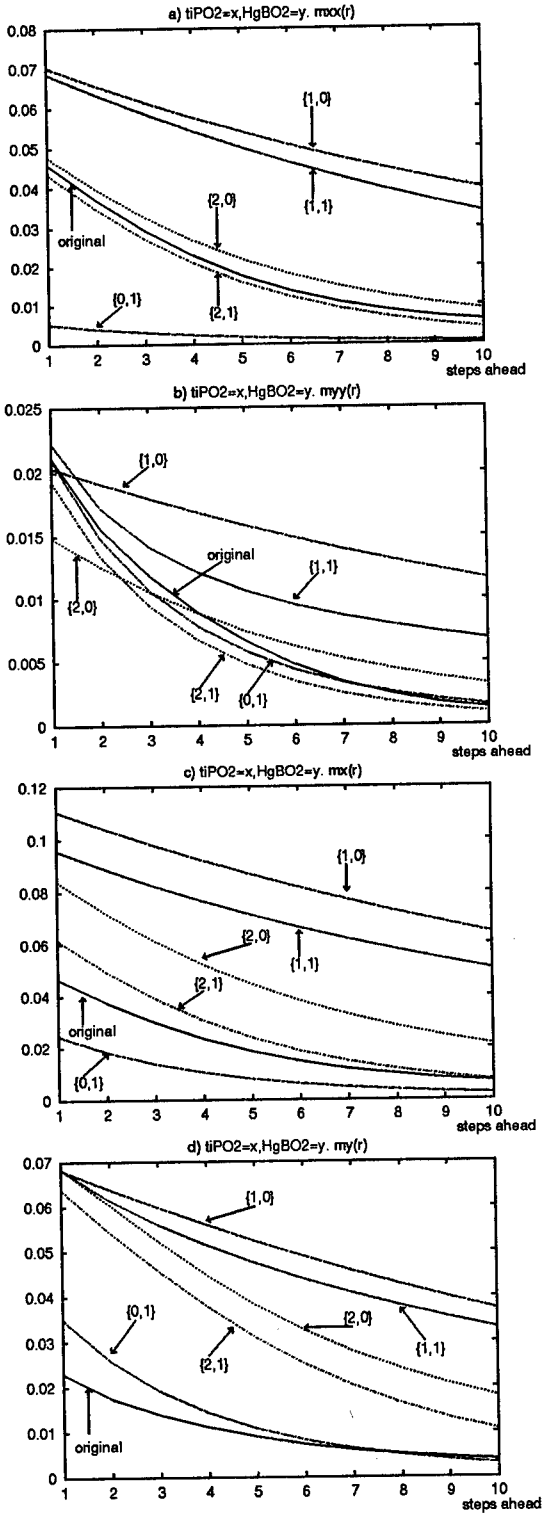


Figure 3: Modelling of HgBO2 and tiPO2 at the time B.

A Multiple-Classifier Architecture for ECG beat classification

Surekha Palreddy, Yu Hen Hu, Vijay Mani, and Willis J. Tompkins

Department of Electrical and Computer Engineering
University of Wisconsin
Madison, WI 53706
hu@engr.wisc.edu

ABSTRACT

In this paper, we investigate the use of modular architecture of multiple clustering based pattern classifiers for ECG beat classification using the MIT/BIH arrhythmia database. The feature space is divided into several regions and individual classifiers are developed for each region separately. Then the outputs of these classifiers are combined using two competing combination rules: a winner decides all method and a distance–based combination method. Experiment results indicated that multiple classifier approach yields better sensitivity and classification rate.

I. INTRODUCTION

ECG beat classification [1, 2, 4, 7, 8, 11, 13, 16, 21, 22, 25, 29] is a difficult pattern classification problem which, despite numerous previous attempts, have not been solved satisfactorily. The difficulties stem from many factors, including large dimension of the feature space, large amounts of training samples, significant overlap between class boundaries and the ever changing morphology (beat shape) with time, just to name a few.

In this study, we report empirical results of performing ECG beat classification to distinguish normal heart beats to those of premature ventricular contraction (PVC) beats using a multi-classifier pattern classification architecture [3, 5, 6, 9, 10, 12, 15, 19, 23, 24, 26, 27, 28]. In this architecture, multiple, separately developed pattern classifiers are combined using a mixture of experts

0-7803-4256-9/97/$10.00

approach. The motivations for studying such an architecture are two folded:

(a) Performance: A modular architecture has the potential to yield better performance than a monolithic classifier architecture. This is a well known fact, and has motivated the study of committee classifiers, as well as mixture-of-expert approaches.

(b) Speed: The modular classifier approach allows parallel development of all component classifiers. Moreover, we choose to divide the training data samples into disjoint data subsets, and therefore further reduces the training time required for individual classifiers.

The component classifier experimented in this study is the Self Organizing Maps (SOM) [14] followed by the Learning Vector Quantization (LVQ) [14] approach. Two types of combination methods are compared: a winner-decides-all method and a distance based weighted average method.

II. COMBINING MULTIPLE PATTERN CLASSIFIERS

In this study, multiple pattern classifiers of the same type are developed independently on different regions of the feature space. This is a divide-and-conquer approach to deal with the large number of training samples. The heuristic is by restricting the training samples of each classifier to a smaller region in the feature space, the classifier can "zoom-in" that region to achieve a better classification result when testing samples fall within that region. This "modular learning" approach potentially offers both performance and speed advantages as stated above. To facilitate the division of the feature space, in this study, initially, the entire training data set is clustered into five clusters with clustering centers $\{\mathbf{C}(i);\ 1 \le i \le 5\}$ using the SOM algorithm. The number of clusters (5 in our experiment) is selected empirically.

Then five LVQ classifiers are developed on each cluster. The LVQ-PAK software (v3.0) is used in this study. Parameters are selected according to the default setting. We denote the output of these classifiers as $\mathbf{y}(i,\mathbf{x})$, with respect to an input feature vector $\mathbf{x}$ for $1 \le i \le 5$. $\mathbf{y}(i,\mathbf{x}) = 1$ if $\mathbf{x}$ is a normal ECG beat, and $= -1$ if $\mathbf{x}$ is

deemed to be a PVC beat. The output of these five classifiers will then combined together using two different combination methods:

(a) Winner decides all method

For each testing feature vector **x**, compute the distance from **x** to the clustering center **C**(i) of the training samples assigned to the i-th classifier: $d(\mathbf{x},i) = \|\mathbf{x} - \mathbf{C}(i)\|^2$. Find i* such that

$$d(\mathbf{x},i^*) \leq d(\mathbf{x},i) \quad \text{for all i.} \tag{1}$$

Then the i*-th classifier is designated as the winner, and it's output is assigned to the output of the combined classifier for **x**. That is,

$$y(\mathbf{x}) = y(\mathbf{x}, i^*) \tag{2}$$

The winner decides all method is a direct result of partitioning the feature space into disjointed regions and training independent classifiers on each region. Consequently, if **x** falls within that region, the corresponding classifier must give the most accurate result.

(b) Distance Based Classifier Output Weighting

In the distance based classifier output weighting method, we compute the combined output as a linear combination of all outputs of the five modular classifiers, and then threshold the output. That is,

$$y(\mathbf{x}) = T[\sum_{i=1}^{5} w(i)y(\mathbf{x},i)]; \quad \sum_{i=1}^{5} w(i) = 1 \tag{3}$$

where T[x] = 1 if $x \geq 0$, and −1 if $x < 0$ is a threshold operator. In order words, since y(**x**,i) = 1 or −1, eq. (3) states that the combined output is obtained by a weighted voting of the component classifiers. The question is how to determine the weighting w(i).

To find an exact solution to such a problem is extremely complicated. Instead, we opt for an approximated solution as follows: We assume that each y(**x**, i) is an independent estimate of

the target value t(**x**), with a variance $\sigma_i^2(\mathbf{x}) = \alpha\, d(\mathbf{x}, i)$ where $\alpha > 0$ is a constant. Denote

$$z(\mathbf{x}) = \sum_{i=1}^{5} w(i)y(\mathbf{x},i) \tag{4}$$

Our goal in choosing $\{w(i)\}$ is to minimize the variance of z(**x**). This constrained optimization problem can be solved using Lagrange multiplier method which leads to the solution

$$w(i) = \frac{K}{d(\mathbf{x},i)}, \quad \text{where } K = \sum_{i=1}^{5} \frac{1}{d(\mathbf{x},i)} \tag{5}$$

which yields a variance of z(**x**)

$$\text{Var}\{z(\mathbf{x})\} = K = \sum_{i=1}^{5} \frac{1}{d(\mathbf{x},i)} \tag{6}$$

This solution leads to the second distanced based weighted voting method to combine classifier outputs.

III ECG BEAT CLASSIFICATION

ECG signals are measured from electrical leads (electrodes) attach to human body at specific locations. Depending on applications, there are 12-lead system for short duration (30 seconds) monitoring, or 3-lead system used for the same purpose. There are also longer term monitoring conducted using fewer electrodes. An ECG recording consists of a sequence of spiky ECG "beats" each represents one contraction of the heart. The entire episode of one heart beat is characterized by a sequence of 5 "complexes", named as P, Q, R, S, and T. Based on the relative position (timing) of these complexes, the shape (height and width), an ECG beat can be categorized into one of many "labels", and a segment of ECG beats may also demonstrate certain rhythm. By analyzing the type of ECG beats, and accompany rhythms, a trained electro-cardiologist is

able to diagnose probable causes of anomalies in patient's heart, so that appropriate treatment can be administered.

ECG beat classification is a difficult task because there is no such thing as a standard ECG beat template. Every healthy person has a slightly different shape of "normal" ECG beats. The rhythm and shape of the beat also vary with respect to time. Sometimes it is not only the shape of an individual beat, but its relative location of appearance in a stream of ECG beat determines what type of this beat might be.

The annotated ECG records available from the MIT/BIH (Massachusetts Institute of Technology and Beth Israel Hospital) arrhythmia database [17] have been used in this study. This database has 48 records, each 30 minutes in length. The data were recorded in two channels (modified limb lead II and modified lead VI) of surface ECGs from long-term Holter recorders. They represent a variety of waveforms, artifacts, complex ventricular, junctional, and supraventricular arrhythmias, and conduction abnormalities. Data from 33 of the 48 records which contain normal beats and PVCs were used for this study. Classifiers were developed and evaluated using subsets of data from channel 1 of these 33 records sampled at 360 Hz.

Accompanying each record in the database is an annotation file in which each ECG beat has been identified by expert cardiologists. These labels are referred to as 'truth' annotations and are used in developing the classifiers and also to evaluate the performance of the classifiers in the testing phase. Data are extracted as one feature vector for each of the beats in all the selected records. Each vector includes one of the two possible labels according to the AAMI recommended practice. Each feature vector for has 9 elements. The first four feature elements are temporal parameters. Temporal parameters such as the R–R intervals are calculated as the time difference between the two consecutive QRS peaks. Temporal features are the R–R interval between the current beat and the previous beat (RR1), between the previous beat and the one before it (RR0), between the current beat and the next beat (RR2), and the ratio of RR1 and RR2. These features are extracted for each individual beat in the database. A ratio of RR1 to RR0 provides an indication of an abnormal timing sequence and helps in identifying an abnormal beat. The next 5 feature elements are extracted based on

morphology. Detailed descriptions of these features can be found in [20].

IV. EXPERIMENTS AND RESULTS

For ECG processing, four basic statistics are calculated according to AAMI (American Association of Medical Instrumentation) recommendation [18]: true positives (TP), false positives (FP), true negatives (TN) and false negatives (FN). This follows a detection scenario. True positive (TP) means a true event of PVC has been successfully detected. False positive (FP) gives number of false alarms, and false negative is the count of missed PVC beats. TN and FN can be similarly defined. Based on these statistics, three performance criteria, sensitivity (Sens), Specificity (Spec) and Positive Predictivity (PP) are computed for each method. *Sensitivity*: (Sens = TP/ (TP + FN)) is the fraction of real events that are correctly detected; *Specificity:* (Spec = TN/ (TN + FP)) is the fraction of false events detected as false events; and *Positive Predictivity:* (PP = TP/(TP + FP)) is the fraction of detection that are true events. Generally, one would want all three criteria approach unity. But often trade-offs must be made. Among these three, sensitivity is considered most critical, with specificity and positive predictivity with decreasing importance. This is because missing a life-threatening ECG beat is considered more serious than a few false alarms which can later be screened out manually. We also compute classification rate C_rate = (FP+FN) / (TP + FP + FN +TN).

In order to improve the reliability of the results, while coping with the large volume of data samples, we use the three-way cross validation method to create three different data sets from the same population: We partition the original ECG data randomly into three approximately equal sized subsets. (Note that this is NOT the clustering partitioning mentioned in section II). Then we combine two of the three subset to make a training data set, and use the remaining one as the testing set. This is rotated among these three subsets so that each subset becomes the testing set exactly ones. All the experiments are repeated three times (trials) with each time applied to one of these three different partitions.

To compare the performance of the multiple classifier approach, we also conducted a baseline experiment using a single

SOM+LVQ classifier to classify the entire ECG data set. The results are summarized below:

Trials	TP	FP	FN	TN	Sens	Spec	PP	C_ rate
1	1944	130	218	22436	89.92	99.42	93.73	98.59
2	1824	82	387	22434	82.50	99.64	95.70	98.10
3	1874	69	365	22419	83.70	99.69	96.45	98.24
Avg	5642	281	970	67289	85.33	99.58	95.26	98.31

Table 1. The results of the classifiers developed using clustering algorithms SOM and LVQ and evaluated using the nearest neighbor approach.

Next, we use the multiple classifier approach, with the winner decides all combination rule. The results are summarized below:

Trials	TP	FP	FN	TN	Sens	Spec	PP	C_ rate
1	1978	207	183	22358	91.53	99.08	90.53	98.42
2	2054	163	156	22353	92.94	99.28	92.65	98.71
3	2083	267	156	22220	93.03	98.81	88.64	98.29
Avg	6115	637	495	66931	92.51	99.06	90.57	98.47

Table 2. The results of 'winner decides all' approach in evaluating the modular networks developed by dividing the input space into 5 regions.

Finally, we use the distanced based weighted voting combination method and the results are summarized in Table 3.

Trials	TP	FP	FN	TN	Sens	Spec	PP	C_ rate
1	2018	218	143	22347	93.38	99.03	90.25	98.54
2	2104	204	106	22312	95.20	99.09	91.16	98.75

3	2109	251	130	22236	94.19	98.88	89.36	98.46
Avg	6231	673	379	66895	94.27	99.00	90.25	98.58

Table 3. The performance of the classifiers evaluated using the distance based approach.

From above table, we observed that compared to a single classifier approach, both multi-classifier combination methods yield higher sensitivity (94%, 93% vs. 86%), while sacrifice somewhat on the positive predictivity. The overall classification rate is also improved from 98.31% to 98.47% and 98.58%.

References

[1] Bortolan, G., Degani, R., and Willems, J. L., "Design of neural networks for classification of electrocardiographic signals," *Proc. Annual Intl. Conf. IEEE Eng. Med. & Biol. Soc.*, no. pp. 1467–1468, 1990.

[2] Cabello, D., J. M. Salceda, S. Barro, R. Ruiz, and J. Mira, "Statistical techniques for diagnosis of ventricular arrhythmias," *Revista de Informatica y Automatica*, vol. 22, no. 4, pp. 45-52, 1989.

[3] Drucker, H., C. Cortes, L. Jackel, Y. LeCun, and V. Vapnik, "Boosting and other ensemble methods," *Neural Computation*, vol. 6, no. 6, pp. 1289-1301, 1994.

[4] Habboush, I., G. B. Moody,and R. G. Mark, "Neural networks for ECG compression and classification," in Proc. *Proceedings. Computers in Cardiology*, pp. 185-188, 1991.

[5] Hansen, L. K., and P. Salamon, "Neural network ensembles," *IEEE Trans. on PAMI*, vol. 12, no. 10, pp. 993-1001, 1990.

[6] Ho, T. K., J. J. Hull, and S. N. Srihari, "Decision combination in multiple classifier systems," *IEEE Trans. on PAMI*, vol. 16, no. 1, pp. 66-76, 1994.

[7] Hu, Y. H., W. J. Tompkins, J. L. Urrusti, and V. X. Alfonso,, "Applications of artificial neural networks for ECG signal detection and classification,," *Journal of Electrocardiology*, no. pp. Accepted, in press, 1994.

[8] Hu, Y. H., Palreddy, S., and W. J. Tompkins, "Patient Adaptable ECG Beat Classification using Mixture of Experts,"

in *Neural Network for Signal Processing V*, Ed(s)., IEEE, 1995.

[9] Jacobs, R., "Method for combining experts' probability assessments," *Neural Computation*, vol. 7, no. 5, pp. 867-888, 1995.

[10] Jacobs, R. A., M. I. Jordan, S. Nowlan, and G. E. Hinton, "Adaptive mistures of local experts," *Neural Computation*, vol. 3, no. pp. 79-87, 1991.

[11] Jenkins, J. M., . and Arzbaecher, R., "On-line computer pattern recognition and classification of ventricular and supraventricular arrhythmias," in *Progress in Electrocardiology, Macfarlane*, Ed(s)., Glasgow: Pittman Medical, 1979.

[12] Jordan, M. I., and R. A. Jacobs, "Hierarchical mixtures of experts and the EM algorithm," *Neural Computation*, no. pp. 1993.

[13] Klingeman, J., and Pipberger, H. V., "Computer classification of electrocardiograms," *Comp. Biomed. Res.*, vol. 1, no. pp. 1, 1967.

[14] Kohonen, T., "The Self-Organizing Map," *Proc. IEEE*, vol. 78, no. 9, pp. 1464-1480, 1990.

[15] Krogh, A., and J. Vedelsby, "Neural network ensembles, cross validation and active learning," in *Advnaces in Neural Information Processing Systems 7*, Ed(s)., Cambridge MA: MIT Press, 1995.

[16] Linnenbank, A. C., Groenewegen, A. S., and Grimbergen, C. A.,, "Artificial neural networks applied in multiple lead electrocardiography: Rapid quantitative classification of ventricular tachycardia QRS integral pattern," *Proc. Annual Intl. Conf. IEEE Eng. Med. & Biol. Soc.*, no. pp. 1461-1462, 1990.

[17] Mark, R., and G. Moody, "MIT-BIH Arrhythmia Database Directory", MIT, 1988.

[18] Mark, R. and R. Wallen, "AAMI Recommended Practice: Testing and Reporting Performance Results of Ventricular Arrhythmia Detection Algorithms", Association for the Advancement of Medical Instrumentation, AAMI ECAR-1987, 1987.

[19] Meir, R., "Bias, variance, and the combination of estiamtors: the case of least linear squares," in *Advances in Neural Information Processing Systems 7*, Ed(s)., Cambridge, MA: MIT Press, 1995.

[20] Palreddy, S., "ECG Beat Classification," Ph. D. Dissertation, Univ. of Wisconsin-Madison, 1996.
[21] Pedrycz, W., G. Bortolan, and R. Degani, " Classification of electrocardiographic signals: a fuzzy pattern matching approach," *Artificial Intelligence in Medicine*, vol. 3, no. 4, pp. 211-226, 1991.
[22] Pipberger, H. V., A. S. Berson, J. D. Klingeman, and C. D. Batchlor, "Diagnostic classifications of orthogonal electrocardiograms and vectorcardiograms," in Proc. *International Vectorcardiogram Symposium*, pp. 157-163., 1971.
[23] Seung, H. S., M. Opper, and H. Sompolinsky, "Query by committee," in *Proc. 5-th workshop on computational learning theory*, Ed(s)., San Mateo, CA: Morgan kaufmann, pp. 287-294, 1992.
[24] Tresp, V., and M. Taniguchi, "Combining estimators using non-constant weighting functions," in *Advances in Neural Information Processing Systems 7*, Ed(s)., Cambridge MA: MIT Press, 1995.
[25] Tsai, Y. S., Hung, B. N., and Tung, S. F., "An experiment on ECG classification using back-propagation neural network," *Proc. Annual Intl. Conf. IEEE Eng. Med. & Biol. Soc.*, no. pp. 1463-1464, 1990.
[26] Tumer, K., and J. Ghosh, "Error correlation and error reduction in ensemble classifiers," *Connection Science, special issue on combining neural networks*, no. pp. (to appear), 1996.
[27] Wolpert, D. H., "Stacked Generalization," *Neural Networks*, vol. 5, no. 2, pp. 241-259, 1992.
[28] Xu, L., and M. I. Jordan, "EM learning on a generalized finite mixture model for combining multiple classifiers," in Proc. *Proc. World Congress on Neural Networks*, Portland, OR, vol. IV, 1993.
[29] Yeap, T. H., Johnson, F., and Rachniowski, "ECG beat classification by a neural network," *Proc. Annu. Int'l Conf. IEEE Eng. Med. & Biol. Soc.*, no. pp. 1457–1458, 1990.

Applying Neural Networks to Adjust Insulin-Pump Doses

Fidimahery Andrianasy Maurice Milgram

Laboratoire PARC, Tour 66, 2ème tage,
Université Paris 6, 4 Place Jussieu, 75 005 Paris, France
{fidi,milgram}@robo.jussieu.fr

Abstract. Programming appropriate insulin-dose levels is a common diabetic pump-user problem. We developed a neural-network advisory system that suggests the appropriate next-time insulin dose based on short historical discontinuous blood-glucose measurements and insulin doses settings. Diabetologists' high level decision taking process have been succesfully learned. Our data base consists of 25000 recorded data from 747 insulin-pump users under medical supervision. The *efficient data* concept is introduced. Training with *efficient learning data* allowed to achieve very good generalization. A portable neural-network controlled insulin-pump device is beeing designed. A complete insulin advisory system including our algorithm is currently under clinical test. Preliminary results demonstrate that the performances of the neural-networks are equivalent to those of the physician.

1 Introduction

An insulin pump is a miniature pump which delivers a continuous supply of insulin (Basal rate), with extra insulin administered for meals (Prandial or Bolus rate) to an insulin-dependant diabetic (IDD). This is the best known system that can closely mimics the body's pancreas normal release of insulin. Unfortunately, blood-glucose (BG) levels have to be monitored before one could make good adjustments in insulin, food, and exercises in response to those glucose test results. BG variations depend on several factors and vary with time. Deciding the amount of injection is a difficult task because morphology, future physical activity, time of meal, meal content, present glucose concentration and results of the previous day have to be taken in account. Moreover, injected insulin acts with delay and its efficiency reduces as BG level gets higher.

Our paper deals with the specific problem of how to predict accurately the insulin dosage of an insulin-pump . Diabetologists' knowledge are generally in heuristic form. The medical staff at Strasbourg's Hospital applies a typical two phases scheme in order to control the BG level of an IDD patient under pump treatment: (1) at the beginning of each day, the diabetologist prescribes an injection rate profile for the next 24 hours, taking into account the present BG concentration and the results of previous day injections; (2) during the

0-7803-4256-9/97/$10.00

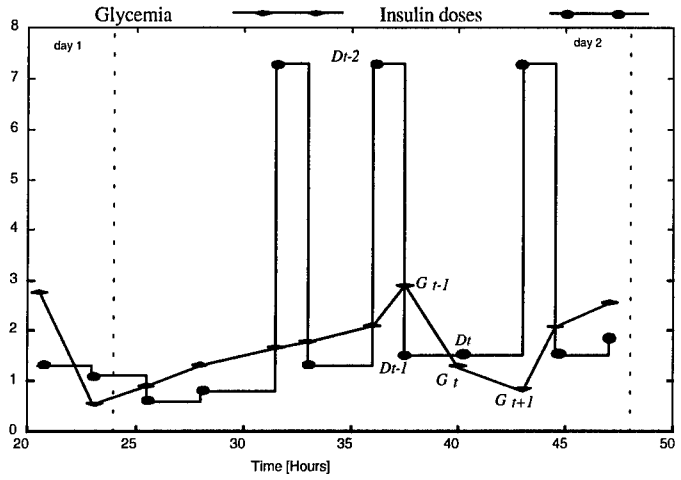


Fig. 1. *A pump–user glycemia responses to insulin doses (pump settings flow rates). Note that pump settings are constant between successive adjustments and the discontinuous measurements are not taken at fixed intervals. Given historical data of a specific patient, the problem is to find appropriate doses $D(t)$ that would induce "normal" next–time glycemia levels $G(t+1)$. "Efficient facts" are selected based on the quality index $|G(t+1)-1|$.*

next 24 hours, the paramedical staff monitors and fine tunes the insulin levels so as to stay into the ideal BG target (i.e.$1g/l$). Ten measurements and tuning per 24 hours are then performed.

A preliminary inspection of the data shows a strong correlation between the profiles of insulin advisories for two consecutive days. We propose to use a neural model to predict the next-time insulin level based on short history (previous hours) and same–time of the previous-day data. Our aim was to extrapolate from physician's experiences hidden in the available clinical data.

Neural-network approach has been successfully applied to various areas of medicine, such as diagnostic aides, biochemical analysis, image analysis, and drug development. But, no known results have been announced on the use of backpropagation networks to drive an insulin-pump [1] especially for the *discontinuous* measurements and infusion. Lakatos *et al.* tried to simulate the specialists' reasoning by predicting first the BG with a neural network; they had to built another network using one-hour time resolution to ensure continuous data consideration and cubic spline interpolation to generate BG profile [2].

Intravenous glucose controlled insulin infusion was achieved until now by using continuous BG measurements [3] [4], which does not make possible a routine use in the treatment of diabetic patients. Meanwhile, numerous studies have been conducted on closely related fields. Much of the efforts

were aimed at the prediction the BG level and the identification of the glucose metabolism [5], [6]. Blumenfeld[7] for example described networks that when presented with the serum glucose and pump settings at time steps t and $t+1$, are capable of predicting the serum glucose, and suggesting the pump setting at time $t+2$. Unfortunately, many theoretical and practical obstacles remain since ideally the network used should interpret all data as *a continuous stream* and all measurements must be taken at *fixed intervals*. Most of these works rely upon classical signal processing techniques.

The problem of insulin-advisory-strategy modelling seems to be less hard to tackle than the glucose metabolism identification one. Insulin flow pattern and human blood-glucose response are highly time-dependent. In our case, measurements are not taken at fixed intervals (called here "periods") even though they were performed at fixed time. We suggest using a multi-network architecture system, where each neural network is dedicated to one period of time. The multi-networks architecture provides an elegant way to resolve the time-dependency problem since, training a neural network under these conditions amounts to finding a static function of its inputs.

On these assumptions, it is the belief of the authors that standard back-propagation neural networks are sufficiently powerful tools for the problem of predicting the insulin doses levels, provided enough complexity of the neural model, i.e. enough number of neurons are used in the hidden layer of each neural network. The "efficient data learning" concept was applyed to enforce the fact that only the best physicians' decisions according to a given quality index, are taken in account. One must bear in mind, that the proposed system requires a good initialization scheme at the very first time of operation when no previous data are available.

2 NEURAL-NETWORK ARCHITECTURE

The goal of this study is to build a neural network model of the empirical rules devised by health care teams for choosing the right insulin dosage, with neural networks system. The objective is to maintain the BG level at the constant value of $1g/l$. Our database come from unpublished data from 747 patients under medical pump-treatment. Daily records of the patients BG and the insulin level infusion have been collected. There are 10 records per 24 hours corresponding to 10 fixed measurements times: $[h_1, ..., h_{10}] = [1h30, 4h00, 7h30, 9h00, 12h00, 13h30, 16h00, 19h00, 20h30, 23h00]$. We suppose that 11 records are already available, that is, the patient have been monitored for at least 24 hours. The case of the 11 first predictions will be treated in detail within a next publication.

The meal time hours $\{h_3, h_5, h_8\}$ are distinguished from the 7 remaining basal hours. The entire raw data set consists of 25,000 facts. In addition to the current and next time BG and insulin dosage, each record includes the time of day, the age, the weight(W), the meal contents and the Bmi factor. The Bmi factor is defined as Bmi $= W/H^2$ where H is the patient's height.

Throughout this paper, if t is the current time ($t = h_i \in \{h_1, ..., h_{10}\}$), then $(t+1)$ refers to the next canonical time, h_{i+1} with $h_{10+1} = h_1$.

Symbol	Range	Input features
G_t	$[-1,+1]$	The current blood-glucose level
G_{t-1}	$[-1,+1]$	The $(t-1)$ blood-glucose level
G_{t-2}	$[-1,+1]$	The $(t-2)$ blood-glucose level
S	$[0,+1]$	Extra sugar (during hypoglycemia)
I	$[0,+1]$	Extra (flash) injection (during hyperglycemias)
G_{t-10}	$[-1,+1]$	The previous day blood-glucose level
G_{t-9}	$[-1,+1]$	The resulted previous day blood-glucose level
Dn_{t-10}	$[-1,+1]$	The previous-day pre-normalized dose
D_{t-10}	$[-1,+1]$	The previous day non-normalized dose
Dm	$[-1,+1]$	The mean of Basal injection since $t = 1$
D_{t-1}	$[-1,+1]$	The previous time $(t-1)$ injection
D_{t-2}	$[-1,+1]$	The injection at time $(t-2)$

Table 1. Input features.

The model has been decomposed into 10 backpropagation neural networks [8] parts, named nn_{hi}. Each neural networks have been trained separately using data associated to each canonical hours. Only cases in which insulin prescriptions were *efficient* were included into each training set. *Efficient learning data* regroups the subset of the learning facts that have induced a *normal* BG level, i.e. $0.9g/l \leq BG \leq 1.3g/l$. Only the 10-th and above facts for each patient are taken into account because the model need the previous day values. The number of features (12) sets the size of the networks input layer (see Table1)

Let $\alpha(t)$ (called the α-factor), be the linear regression coefficient between the basal rate vector $[D(1), ..., D(t)]$ and the corresponding standard basal-rates vector $[D_{\mathrm{std}}(1), ..., D_{\mathrm{std}}(t)]$. The $D_{\mathrm{std}}(h_i)$ ($i \in \{1, 2, 4, 6, 7, 9, 10\}$), are standard basal-rates on time t i.e. $D_{\mathrm{std}}(t) = D_{\mathrm{std}}(h_i \equiv t)$. In this paper $D_{\mathrm{std}}(h_i) = [0.8, 0.8, 1.2, 1.6, 1.6, 1.3, 1.3]$. These are provided by diabetologists. The following formula was used to evaluate the α-factor,

$$\alpha(t) = \sum_{i=1,\ ...\ ,t} (D_{\mathrm{std}}(i)D(i)) \ / \sum_{i=1,\ ...\ ,t} (D_{\mathrm{std}}(i))^2$$

The injection dose $D(t)$ was normalized using the coefficient $\alpha(t)$ yielding $D_n(t) = D(t) - \alpha(t) . D_{\mathrm{std}}(t)$. This pre-normalization was only applied to the previous-day injections $D(t-10)$.

The output is the current insulin injection rate D_0, normalized to the range $[-1,+1]$. The most successful backpropagation network architecture had 12 inputs, 4 cells in the hidden layer and one element in the output layer $(12:4:1)$. Then the entire system is composed of $10 \times (12:4:1)$ networks. The trained networks have been tested using unseen patterns consisting of about one-half of the entire data set.

Neural Network	Canonical hours	Mean Errors	
		selected data	raw data
nn_{h1}	01*h*30	0.1540	0.1594
nn_{h2}	04*h*00	0.1152	0.1219
nn_{h3}	07*h*30	0.1085	0.1046
nn_{h4}	09*h*00	0.0892	0.1066
nn_{h5}	12*h*00	0.0945	0.1273
nn_{h6}	13*h*30	0.0713	0.0886
nn_{h7}	16*h*00	0.0641	0.0841
nn_{h8}	19*h*00	0.0966	0.1158
nn_{h9}	20*h*30	0.0896	0.1005
nn_{h10}	23*h*00	0.0616	0.0674

Table 2. *Each row exhibits the performance of neural networks nn_{hi}, $i = 1, ..., 10$ associated to canonical hours h_i. Relatively small errors in the third column show that the networks generalize well on unseen data. The fourth column displays relative-errors when the system is tested against non-filtered data. Error rates are similar to those of the previous case. Larger error rates during the early hours of the day suggests using neural networks with larger hidden layer for these periods in order to handle the increased complexity of the model.*

3 Results

Results on the test set are expressed in terms of mean of relative error $e_m = \sum_k |(D_0^k - \tilde{D}_0^k) \ / \ D_0^k|$, where $\tilde{D}_0^k$ is the network's output for the input pattern k and D_0^k is the desired output. The training were stopped after 10000 presentations (`learning rate` $= 0.0002$) and good performances were obtained as observed in Table 2.

Each row of the table corresponds to a neural network specialized in the prescription of insulin dose at a specific canonical hour. The second column shows networks' performances in generalization when only *efficient* facts were selected. The errors in the third column were obtained using raw unfiltered data. As could be expected, the test conducted with the efficient facts were better than those with the raw data. The relatively small differences between these two tests clearly shows that the trained networks have successfully extracted the actual rules devised by the diabetologists.

4 Conclusions

Pump therapy is for people who have insulin-dependent diabetes, who are able to monitor their blood glucose values and operate the pump themselves. But, deciding the amount for insulin injection is very difficult. We proposed a neural network-based system that predicts the appropriate next insulin doses level for an insulin-pump, given short historical discontinuous-measurements of BG levels and insulin doses.

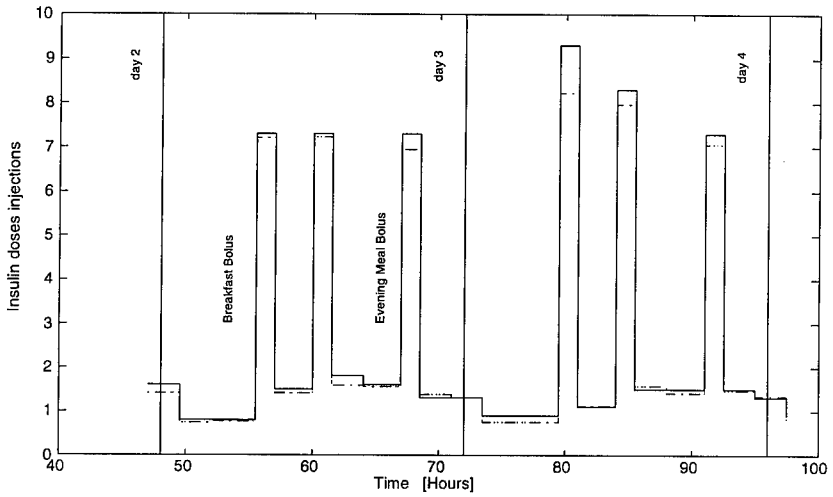


Fig. 2. *Typical* DEBNET *predictions (dashed lines) vs actual insulin injected (solid lines.) Outputs from the second day only have been plotted.*

Our approach allowed us to model the empirical rules devised by health-care teams when computing the insulin level infusion of an insulin-pump device. Neural networks trained with *efficient data* provided good accuracy in predicting appropriate insulin levels. The behavior learned from these facts extrapolates very well for unseen data.

Results suggest that:

- finding a viable control strategy in the particular case of blood-glucose level control is feasible even if the available observed data is discontinuous in nature and sampled irregularly over time; in the past, intravenous glucose controlled insulin infusion was only achieved by using continuous BG measurements;
- standard back-propagation neural networks are powerful enough to model the heuristic rules devised by experienced health-care team when to decide the amount of insulin doses applied at each specific circumstances;
- the normal body insulin release seems to be a strongly time-dependant process, with a varying time-constant depending on the period of the day; this time-constant roughly ranges from 10 to 15 minutes around meal-time and more than 3 hours during the sleeping period of the night;
- the multi-networks approach appears to be the most natural way to handle the variable time-constant problem since the number of different processes to be modelled remains low;
- training back-propagation neural networks with *efficient-data* seems to enhance generalization performances.

A software prototype called DEBNET was put under clinical test. Preliminary reports are very encouraging [9] since the health-care team almost never had to intervene during its operation (see Figure 2). In this version DEBNET used a simple algorithm to initialize the system Due to the lack of space we will enter in the details of the implemented initialization scheme within a next publication. A portable pump controlled by DEBNET is also being designed. Such device (insulin-pump + DEBNET) would help hospitals cut costs by providing faster and accurate prescriptions with fewer costly specialists.

REFERENCES

[1] P. Ladyzinski, J. Wojcicki, and J. Blachowicz, "Prediction of the insulin doses during long-term intensive insulin treatment in diabetic pregnant women using fuzzy logic technique," in **3rd Symposium on Comput. In Diabetes**, 1996.

[2] G. Lakatos, E. Carson, and Z. Benyo, "Artificial neural network approach to diabetic management," in **Proc Ann Intl Conf Eng Med and Biol Soc** (C. Morucci, Plonsey, ed.), vol. 14 Part 3, pp. 1010–1011, Oct. 1992.

[3] F. Skrabal, Z. Trajanoski, H. Kontschieder, and P. Kotanko, "Portable system for on-line continuous ex vivo monitoring of subcutaneous tissue glucose using open tissue perfusion," **Med & Biol Eng & Comp**, vol. 33, pp. 116–118, 1995.

[4] S. Andreassen, J. Benn, R. Hovorka, K. Olesen, and E. Carson, "A probabilistic approach to glucose prediction an insulin dose adjustment," **Computer Methods and Programs in Biomedecine**, no. 41, pp. 153–165, 1994.

[5] T. Deutsch, E. Lehmann, E. Carson, A. Roudsari, K. Hopkins, and P. Sönksen, "Time series analysis and control of blood levels in diabetic patients," **Computer Methods and Programs in Biomedecine**, no. 41, pp. 167–182, 1994.

[6] Z. Trajanoski and P. Wach, "Regularization networks for glucose system identification," in **Proc 16th Ann Intl Conf IEEE Eng Med Biol Soc, Baltimore, USA**, pp. 1083–1084, 1994.

[7] B. Blumenfeld, "A connectionnist approach to the recognition of trends in time-ordered medical parameters," **Computer Methods and Programs in Biomedecine**, vol. 32, no. 1, pp. 53–62, 1990.

[8] D. E. Rumelhart, J. L. McClelland, and PDP Research Group, **Parallel Distributed Processing: Exploration in the Microstructure of Cognition**, vol. 1 & 2. Cambridge, MA; MIT Press, 1986.

[9] M. Pinget, M. Milgram, N. Jeandidier, F. Andrianasy, S. Boivin, and Ph.Friess, "Intravenous glucose controlled insulin infusion using discontinuous capillary blood glucose measurements by means of neural network," **9th Biennal ISGIID Meeting**, 1996. Accepted for publication.

MR BRAIN IMAGE CLASSIFICATION BY MULTIMODAL PERCEPTRON TREE NEURAL NETWORK

Iren Valova, Yukio Kosugi
Interdisciplinary Graduate School of Science and Engineering
Tokyo Institute of Technology, 4259 Nagatsuta, Midori-ku, Yokohama 226, Japan
tel: +81 45 924 5465, fax +81 45 922 5172, email: iren@pms.titech.ac.jp

Abstract

In this paper we propose a multimodal perceptron tree (MMPT) neural network to segment magnetic resonance (MR) images. The architecture consists of simple networks - neurons, hierarchically connected in a tree structure. The latter is built up during training by the adopted depth-first searching technique augmented with choosing the best hyperplane split of the feature subspace at each tree node. This neural network effectively partitions the feature space into subregions and each terminal subregion is assigned to a class label depending on the data routed to it. As the tree grows, the number of training data for each node decreases, which results in less weight update epochs and decreases the time consumption.

The MMPT performance is compared to that of a multilayered perceptron (MLP). The networks are applied to brain MR image segmentation into gray matter/white matter regions.

1. INTRODUCTION

If the analysis of brain MR images could be automated, it would provide accurate and reproducible results as well as relief human experts from time consuming tasks [1]. MRI has been exploited for noninvasive diagnosis as well as for tissue identification for surgical planing and for interpreting other images such as positron emission tomography. In order to apply MR, we often have to introduce a reasonable segmentation technique. Neural networks may provide us with superior solutions for the pattern classification of medical images, than the conventional methods. Neural-network-based segmentation of MR images according to the tissue characteristics has significant meaning for neoplasms diagnosis, 3D display of human organs during surgery simulation and MR - based deconvolution of PET images [2, 3].

In this paper we combine the concept of decision tree classifiers, presented in [4, 5] with the popular network structure of MLP. The basic idea behind the tree

0-7803-4256-9/97/$10.00 ©1997 IEEE

classifiers is to organize the class assignment into a tree search, which is guided by properly chosen questions based on the feature vector. In the conventional systems the node questions are represented as internal nodes of the tree and the class labels - as leaves. In case of many feature vectors, a decision tree will check one attribute at a time which means restricted orientation of hyperplanes. This can also cause the growth of unreasonably large trees [6]. The proposed neurons consider a combination of feature attributes thus greatly simplifying the tree structure.

The MLP has been successfully applied in many image processing tasks. However, it still has drawbacks which make the construction of MLP more of an art than science. The MMPT attempts to eliminate some of the MLP's shortcomings, e.g. the need to specify the architecture in beforehand, choice of activation function in order to speed up the convergence and to avoid the notorious local minima [7, 8].

Other related method exists in the field of speech processing [9, 10]. That method is discussed in Section 4 of this paper in comparison to the proposed method.

2. MMPT LEARNING ALGORITHM

The MMPT is grown through training and therefore the number of units need not be specified in beforehand. At every tree node, a combination of feature vectors is used to form a splitting hyperplane.

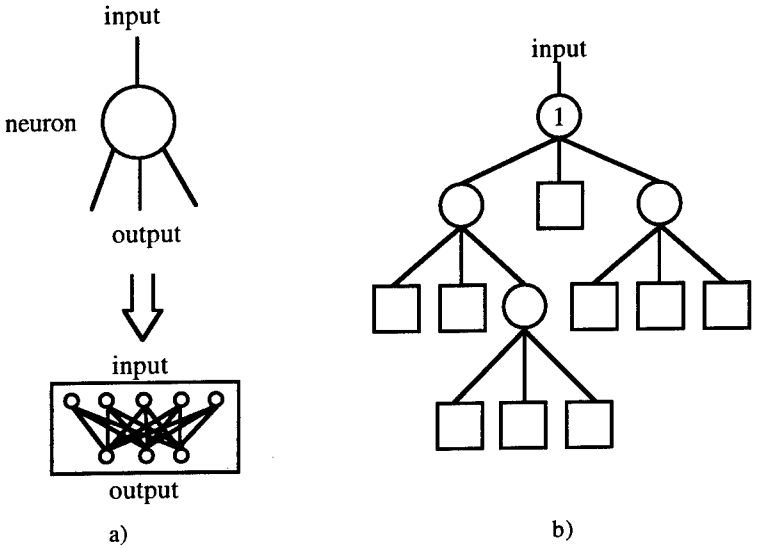


Fig.1 Multimodal Perceptron Tree neural network architecture: a) architecture of a neuron as a part of MWPT network; b) example of a full grown 3-way MWPT

The tree architecture is shown in Fig.1. A neuron consists of a simple net-

work with no hidden units as illustrated in Fig.1a. A matured tree for 3-way classification is shown in Fig.1b. The squares are leaves, i.e. classes to which the data belong after the training is completed.

The growing procedure starts with training the root node - the #1 in Fig.1b, over all training data. The classes are labeled using binary vectors and the data are classified according to a *winner-take-all* rule:

$$\text{sample} \in \text{class } i \text{ if } f(w_i.x) \geq f(w_j.x), \ \forall j \neq i \tag{1}$$

where **w** is the weight vector, **x** represents the input vector and f is the activation function. After training the nodal neural network, the *winner-take-all* divides the feature space into three convex regions. The training data in each region is assigned to a different child node. If the data represent only one class, then the node turns into a leaf signifying this particular class. In case of handling misclassified data, the training/splitting procedure takes place at the particular child node thus decreasing the number of training exemplars for the next layer of child nodes. In this way the training data set is divided and subdivided in recursive fashion until the leaf nodes representing the actual classes are reached. Therefore the number of training examples decreases with increasing of the tree depth. This leads to reduced time consumption compared to MLP because of the MLP's moving target problem, i.e. the weights are changing at once and each hidden unit sees a continuously changing environment. Instead of moving quickly to assume useful roles in the problem solution, the hidden units engage in many wasted motions.

The neurons apply a sigmoid activation function as,

$$f = \frac{1}{1 + e^{-x}} \tag{2}$$

with the output range (0, 1) and are trained independently of each other by a gradient decent method with respect to the least mean square error (L2 norm)

$$E_i = (t_i - y_i)^2, \tag{3}$$

where t_i is the required output for pattern i and y_i is the actual neuron output. The weights are adjusted according to:

$$w_i^{n+1} = w_i^n - \eta \, y_i (1 - y_i) (t_i - y_i) x_i + \alpha (w_i^n - w_i^{n-1}) \tag{4}$$
$$0 < \eta < 1 \text{ and } 0 < \alpha < 1,$$

where w^n is the weight from the i - th input x_i to the neuron at iteration n, η is the training coefficient and α is so called momentum coefficient. The weight update procedure will be stopped when the average error does not decrease beyond a small threshold level over some time span.

The momentum term (α (w_i^n - w_i^{n-1})) is added to speed up the convergence. The training procedure stops if no further splitting is necessary. During testing the data are fed into the root neuron and directed according to (1) to the respective child-neurons. The ideal response of the perceptrons will be 0 and 1, however, in practice, we set the reaction registration thresholds to 0.1 and 0.9.

This is a description of a general tree growing algorithm. The characteristics which set off the proposed algorithm are the depth-first searching technique and the estimation of the bets fit neuron to split the feature subspace for each particular node. The chosen searching technique is guaranteed to find deep solutions, as it searches every branch to the final classifying split in vertical fashion, which is important in case of binary or small-class (three- to four-way branching) trees and also in cases of highly non-linear separable problems as the intertwined spirals [11]. However, regardless of the type of searching technique, a randomly chosen perceptron may offer poor division of the feature space, thus providing its child-nodes with input information, which is inseparable or which will prolong and make the solution finding complicated [12]. This is the reason why we introduce a measure for estimating the most proper neuron for every decision making node in the tree.

The examination procedure includes the training of the node until receiving its output. The input data set includes one or more input values per sample (Fig.2a), which are the outputs of the parent-nodes for this particular sample. This is valid only for child-nodes, as the root node is taken at random. We then form the function

$$\xi = \sum_{o=0}^{m-1} \left| \sum_{s=0}^{n-1} (y_s - \bar{y}) (E_{o,s} - \bar{E}_o) \right| \qquad (5),$$

where y_s is the output for particular input, $\bar{y}$ is averaged value over all samples, $E_{o,s}$ is the error for sample s at output o, $\bar{E}_o$ is the averaged value over all samples, m is the number of outputs, and n is the total number of training samples. The purpose is to obtain a node with maximum ξ, which will mean maximum magnitude of correlation between a unit's value and the residual error E_o, estimated at output o. Instead of creating and examining node by node we set a pool of candidates with different initial weights and examine the whole pool for the best fit candidate according to the described criterion. When such neuron is found, it is adopted in the network architecture. With deepening of the tree, each node takes one more input, i.e. the input space dimensions are increased, which makes the task of classification easier [13]. A flowchart of the algorithm is shown in Fig.2b. In Table 1 we have shown the results for the intertwined spirals task for the proposed method and the one presented in [9, 10]. No pruning was applied in both cases. With this result it can be concluded that the proposed method demonstrates better classification abilities and therefore the introduced measure and the applied searching technique present better fit method for the purposes of image classification.

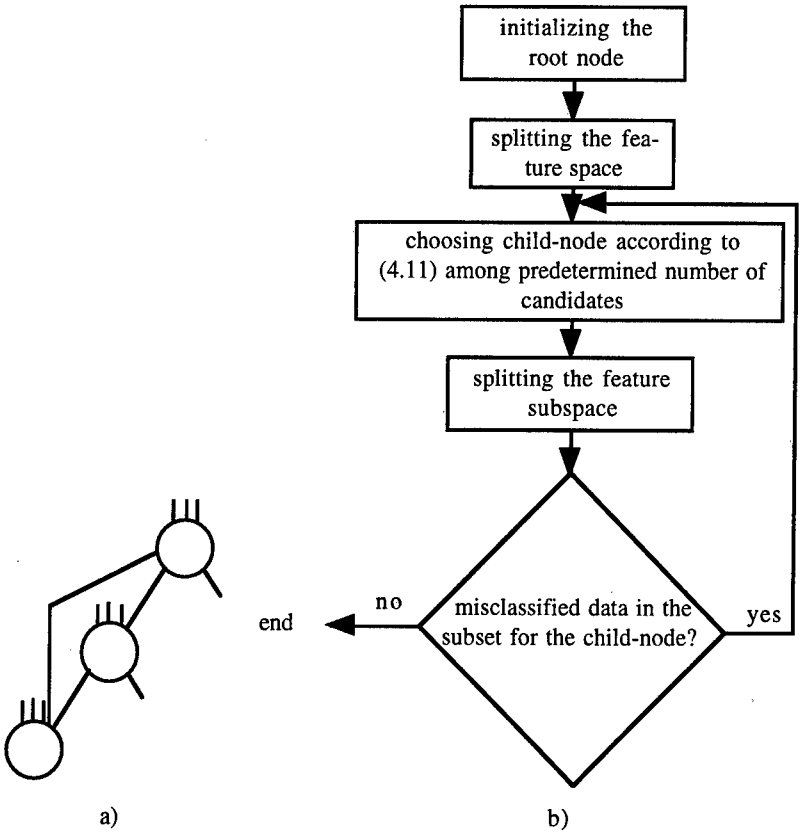


Fig. 2 The tree growing algorithm: a) scheme for applying the measure for choosing best fit neuron; b) flowchart of the method

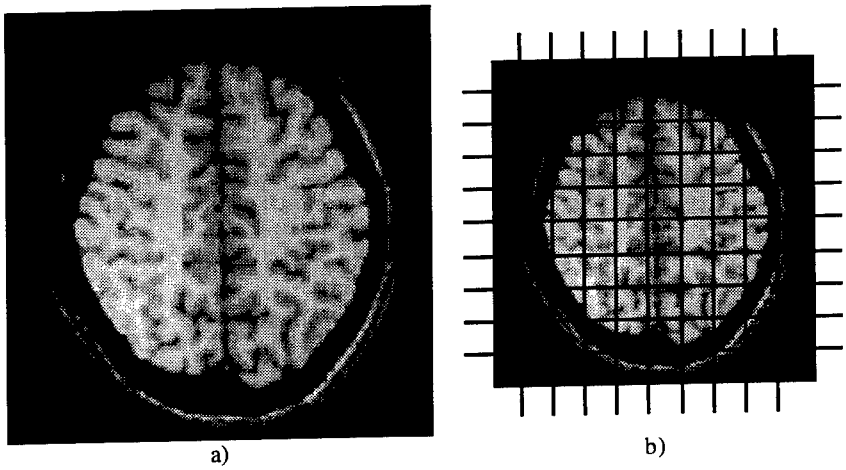
TABLE 1 PERFORMANCE COMPARISON BETWEEN MMPT AND STATE-OF-ART METHOD ON INTERTWINED SPIRALS TASK

	MMPT	Zrida, Mammone [9, 10]
correct classification	88%	63%

3. SEGMENTATION RESULTS

The brain MR image to be segmented is shown in Fig.3a. It is 160 x 160 pixels, 256 shades of gray scale image. The information contained in this image can be divided basically into three classes - the cerebro-spinal fluid (CSF), muscles, gray/white matter. The aim is to recognize the white matter and gray matter re-

gions, thus the result can be used as *a priori* knowledge in the processing of positron emission tomography (PET) images [3].



a) b)

Fig.3 MR image to be segmented: a) MRI gray scale 160x160 pixel; b) partitioned image into 20x20 blocks

The procedure for classification is divided into two steps - in Step 1, we single out the muscles and CSF portion as unimportant information. In order to proceed with Step 1 we partition the image as shown in Fig.3b, which leaves us with 400 blocks to classify as non-brain tissue, border blocks, i.e. containing brain tissue as well as non-brain tissue, and gray/white blocks. We divide the brain tissue in this way, since the data require different input feature vectors to be classified correctly by the neural network.

Step 2 is responsible for the pixel-by-pixel classification into the white matter/gray matter regions. This step is also applied to MMPT network.

The drawback of having two steps is the requirement for more than 97% correct classification at least in Step 1. On the other hand such division of the process allows precise classification of the meaningful information - the brain tissue.

The described process is summarized in Fig.4, where the ellipsoids signify the data subject to further processing, and the rectangles contain the final result to be achieved.

MMPT network with 5 inputs and 3 outputs was implemented in Step 1. The input feature vectors were determined as follows:

-averaged pixel value over a block { avrg = Σ_b b p(b), p(b) is the probability density of pixel value b };

-variance of a block { vrnc = Σ_b (b - avrg)^2 p(b) };

-skewness of a block { skew = [Σ_b (b - avrg)^3] / vrnc$^{3/2}$ };

-Laplacian operator of a block { calculated in 8-neighbourhood according to 3x3 mask };

-maximum pixel value for the block.

The results for MMPT and MLP are given in Table 2 and the resultant image is shown in Fig.5a.

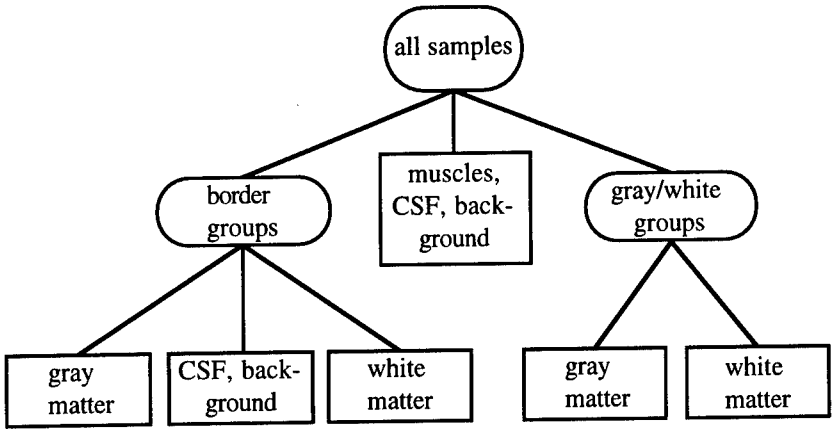
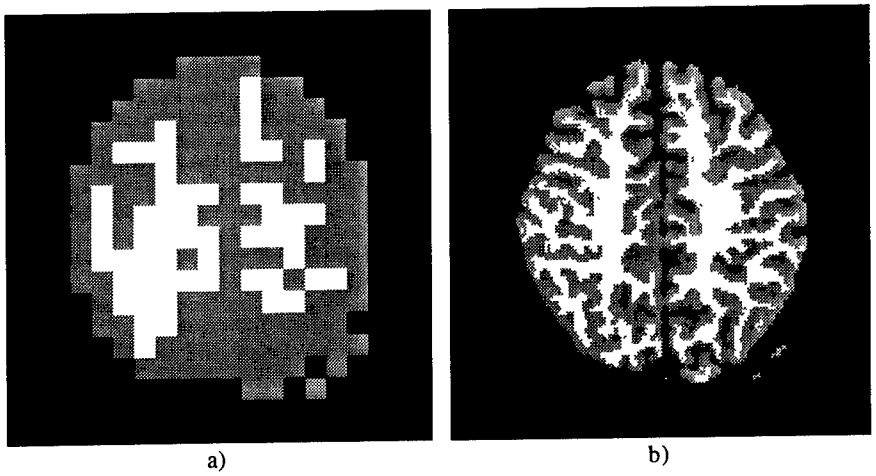


Fig.4 Structure of the segmentation process



a) b)

Fig.5 Results of the two step segmentation procedure: a) Step1 - classification of the groups; b) Step 2 - classification according to the pixel attributes

In Step 2 we implement two MMPTs - one with 4-inputs-3-outputs for the processing of the border blocks and 3-inputs-2-outputs for the gray/white blocks. The inputs for the border blocks are:

-pixel value;

-difference from the highest pixel in the block;

-difference from the highest neighbor within distance 1;

-difference from the next to the highest pixel in the block.

Three outputs for the unimportant information, gray matter pixels and white matter

ones. In case of gray/white blocks we have:

-pixel value;

-difference from the highest pixel in the block;

-difference from the highest neighbor within distance 1;

as input feature vectors as well as 2 outputs for gray matter and white matter classes.

TABLE 2 PERFORMANCE COMPARISON BETWEEN MMPT AND MLP

	% of correct classification		computational complexity (weight updates)	
Classification	MMPT	MLP	MMPT	MLP
Step 1	98.9%	98.5%	511 455	1 512 235
Step 2 - border blocks	90.5%	88%	812 500	2 050 345
Step 2 - gray/white blocks	89%	85.5%	857 140	1 523 010

The results are given in Table 2 and the final segmented image is shown in Fig.5b. The same steps were applied to MLP. The results of MLP segmentation are also presented in Table 2. Obviously the performance of the MLP is slightly worse than that of MMPT. But the practical advantage of MMPT network to MLP is in the dramatically decreased computational cost, which we measure as weight updates required for training. The data is shown in Table 2 and the difference is evaluated in terms of times.

4. DISCUSSION

Although based on decision trees, MMPT does not inherit the major shortcomings of that approach. The proposed architecture offers elegant way of splitting the feature space by hyperplanes not necessarily orthogonal to the axes. A basic feature of the decision trees is that they work with symbolic representation of the input information, which is not suitable for many applications. By applying neural networks, grown as decision tree we allow the decision tree to deal with numerically expressed information and we incorporate the ability of the decision tree to interpret *if-then* rules into the network. The latter is trained based on simple delta-rule, rather than exhaustive search throughout the whole feature set based on calculation of an information criterion.

The presented in [9, 10] method is based on breadth-first searching technique, which builds the tree in horizontal fashion, i.e. level after level; one short-

coming that, it is not guaranteed to find solutions deep into the tree hierarchy and therefore is unsuitable for application in binary or small-class tree cases. The technique also does not perform any estimation of the neuron fitness prior to adopting this node into the tree hierarchy. The authors rely on post-building hyperplane perturbance strategy and tree pruning. The proposed method - MMPT - incorporates depth-first search, i.e. every branch is searched to the final classifying split in vertical fashion. With this approach we also propose an estimation measure for the best neuron to split the feature subspace. The depth-first search, combined with the introduced measure ensures high correct classification rates even for difficult tasks, like the intertwined spiral problem.

The proposed method has been developed for binary trees initially [14]. In this case the procedure takes four processing steps. The MMPT adopts the natural hierarchy of the process (Fig.4) as it allows for three-class segmentation. This reduces the value of the total error, made by the hierarchical processing steps. Here we also introduce more effective measure for estimation of the best fit neuron.

A question that might arise concerns the way to choose the proper input feature vectors for the training. For the discussed examples they were chosen as best combinations out of a set of combinations - combinations of 9 candidates per classification were checked. This was done by cut and try technique. However, if this procedure is done in the light of class-entropy criterion, the process will be more accurate, systematic and speedy [15]. Another option for improving the algorithm is to include not only sigmoid but also Gaussian units. From experiments with artificial patterns we have concluded that Gaussian activation function could decrease the computational cost and better the performance.

5. CONCLUSION

In this work we apply the MMPT neural network for image segmentation problems. The proposed network training algorithm combines depth-first search with the introduced best fit measure, which ensures high correct classification results even when the problem is highly is non-linearly separable. Although it is based on decision tree growing, the MMPT does not resort to the exhaustive search techniques as used in decision trees and offers elegant way of splitting the input feature space due to the freedom in choosing the hyperplane orientation.

The obtained classification rate is comparable to the performance of MLP, but the proposed MMPT has some additional advantages:

- building through training;
- lower computational and structural complexity;
- easier hardware implementation due to smaller number of connections.

However the following remarks should be noted:

- implementing different activation functions in order to reduce the computational complexity and cost;
- implementation of more reliable method for choosing the proper input feature vectors.

REFERENCES

[1] Sammouda R., Niki N., Nishitani H., "Segmentation of Brain MR Images Based on Neural Networks", **IEICE Trans. on Inf.&Syst.**, vol. E79-D, No.4, April, 1996.

[2] Sase M., Kinoshita N., Kosugi Y., "A Neural Network for Fusing the MR Information into PET Images to Improve Spatial Resolution", **Proc. IEEE Int. Conf. on Image Processing**, pp 908-911, 1994.

[3] Kosugi Y., Sase M., Suganami Y. et al., "Dissolution of Particular Volume Effect in PET by an Inversion Technique with the MR-embedded Neural Network Model", **Proc. Brain'95**, S35, 1995.

[4] Quinlan J.R., "Learning efficient classification procedures and their application to chess-end games, in (eds. R.S. Michalski et al.), **Machine Learning**, Morgan Kaufmann, 1983.

[5] Pao Y.H., **Adaptive Pattern Recognition and Neural Networks**, Addison-Wesley Publishing Company, 1989.

[6] Forsyth R., Rada R., **Machine Learning Applications in Expert Systems and Information Retrieval**, Ellis Horwood Ltd., 1986.

[7] Fausett L., **Fundamentals of Neural Networks - Architectures, Algorithms, and Applications**, Prentice Hall, 1994.

[8] Bose N.K., Liang P., **Neural Network Fundamentals with Graphs, Algorithms, and Applications**, McGraw Hill Inc., 1996.

[9] J. E. Stromberg, J. Zrida, A. Isaksson, "Neural Trees: Usin Neural Trees in a Tree Classifier Structure", **IEEE ICASSP - 91**, Toronto, Canada 1991, pp 137-140

[10] A. Sankar, R. Mammone, "Growing and pruning neural tree networks", **IEEE Trans. on Computers**, vol. 42, No 3, March 1993, pp 291-299

[11] K. Lang, M. Witbrock, "Learning to tell two spirals apart", **Proc. Connectionist Models Summer School**, 1988, pp 52-59

[12] R. Mammone (ed), **Artificial NNs for speech and vision**, Chapman & Hall, 1994

[13] N. Nilsson, The mathematical foundations of learning machines, CA: Morgan Kanfmann, 1990

[14] I. Valova, Y. Kosugi, "Brain MR Image Segmentation Implemented on Neural Dichotomizer", **Collection of Papers of the First International Workshop on Advanced Signal Processing for Medical MRI-MRS (ASPM/MRI-MRS)**, Xanthi, Greece, part 3, pp 1-4, 1997

[15] Kosugi Y., Suganami Y. et al, "CCE-Based Index Selection for Neuro Assisted MR-Image Segmentation," **IEEE International Conference on Image Processing**, vol.II, pp 249-252, September 1996, Lausanne, Switzerland.

TEXTURE ANALYSIS AND ARTIFICIAL NEURAL NETWORK FOR DETECTION OF CLUSTERED MICROCALCIFICATIONS ON MAMMOGRAMS

Jong Kook Kim, Jeong Mi Park,* Koun Sik Song,* and Hyun Wook Park
Department of Information and Communication Engineering,
Korea Advanced Institute of Science and Technology,
207-43, Cheongryangri, Dongdaemungu, Seoul 130-012, Korea
*Department of Diagnostic Radiology,
Asan Medical Center, University of Ulsan College of Medicine,
388-1, Poongnap-Dong, Songpagu, Seoul 138-040, Korea

Abstract

Clustered microcalcifications on X-ray mammograms are an important sign in the detection of breast cancer. This paper quantitatively describes the usefulness of texture analysis methods for the detection of clustered microcalcifications on digitized mammograms. Comparative studies of texture analysis methods are performed for the proposed texture analysis method, called the surrounding region dependence method (SRDM), and the conventional texture analysis methods such as the spatial gray-level dependence method (SGLDM), the gray-level run length method (GLRLM), and the gray-level difference method (GLDM). These methods are applied to classify region of interests (ROIs) into positive ROIs containing clustered microcalcifications and negative ROIs of normal tissues. The database is composed of 72 positive and 100 negative ROI images, which are selected from digitized mammograms with a pixel size of 100 × 100 μm^2 and 12 bits per pixel. An ROI is selected as an area of 128 × 128 pixels on the digitized mammograms. A three-layer backpropagation neural network is employed as a classifier. The results of the neural network for texture analysis methods are evaluated by the receiver operating-characteristics (ROC) analysis. From the viewpoint of the classification accuracy and computational complexity, the SRDM is superior to the conventional methods.

0-7803-4256-9/97/$10.00

1. INTRODUCTION

The early detection of breast cancer is the most important factor for reducing breast cancer mortality. It has been empirically recognized that the presence of clustered microcalcifications on X-ray mammograms has been associated with an important sign in the detection of breast cancer, where individual microcalcifications is up to about 0.7 *mm* in diameter and with an average diameter of 0.3 *mm* [1]. Computer-aided diagnosis (CAD) has been of interest to many researchers for the detection of clustered microcalcifications on mammograms [2]-[4].

In this paper, the usefulness of texture analysis methods is quantitatively analyzed for the detection of clustered microcalcifications. The texture analysis methods are the surrounding region dependence method (SRDM) proposed by authors [5], the spatial gray-level dependence method (SGLDM) [6], the gray-level run length method (GLRLM) [7], and the gray-level difference method (GLDM) [8]. The three-layer backpropagation neural network is employed as a classifier. The results of the neural network for these texture analysis methods are evaluated by the receiver-operating characteristics (ROC) analysis [9]. The area under the ROC curve, A_z, is used as a measure of the classification performance.

2. THE SURROUNDING REGION DEPENDENCE METHOD (SRDM)

The SRDM is based on the second-order histogram in two surrounding regions. Let us consider three rectangular windows centered on a current pixel (x,y), as shown in Fig. 1. In Fig. 1, R_1 and R_2 are the inner surrounding region and the outer surrounding region, respectively, and w_1, w_2, and w_3 denote the size of each square region. In this study, w_1, w_2, and w_3 have the values of 3, 5, and 7, respectively. A region of interest (ROI) image is transformed into a surrounding region dependence matrix, which is defined as

$$M(q) = [\alpha(i,j)], \quad 0 \le i \le m,\ 0 \le j \le n \tag{1}$$

where q is a given threshold value, and the values of m and n are the total numbers of pixels of regions R_1 and R_2, respectively. In eq. (1), the element $\alpha(i,j)$ is given as

$$\alpha(i,j) = \#\{(x,y) | c_{R_1}(x,y) = i \text{ and } c_{R_2}(x,y) = j,\ (x,y) \in L_x \times L_y\} \tag{2}$$

where # denotes the number of elements in the set, and $L_x \times L_y$ is the 2-D image space. In eq. (2), the inner count $c_{R_1}(x,y)$ and the outer count $c_{R_2}(x,y)$ on the current pixel (x,y) are defined as follows:

$$c_{R_1}(x,y) = \#\{(k,l) | (k,l) \in R_1 \text{ and } [S(x,y) - S(k,l)] > q\} \quad (3)$$

$$c_{R_2}(x,y) = \#\{(k,l) | (k,l) \in R_2 \text{ and } [S(x,y) - S(k,l)] > q\} \quad (4)$$

where $S(x,y)$ is the image intensity on the current pixel (x,y). In general, the larger the threshold value q is, the more microcalcifications can be missed, whereas the smaller the value q is, the more sensitive the random noise effect is, so that negative ROIs can be classified as positive. The optimal selection of the q value is very important for the classification performance.

It is evident that the surrounding region dependence matrix $M(q)$ contains the textural information of an image. The texture coarseness or fineness of an image can be interpreted as the distribution of the element in the matrix $M(q)$. Especially, the distribution of elements tends to spread near the right and/or the right-lower corner of the matrix for positive ROIs containing clustered microcalcifications. From the spread characteristics of the elements in the surrounding region dependence matrix, we defined four textural features, which were the *horizontal weighted sum* (*HWS*), the *vertical weighted sum* (*VWS*), the *diagonal weighted sum* (*DWS*), and the *grid weighted sum* (*GWS*) [5].

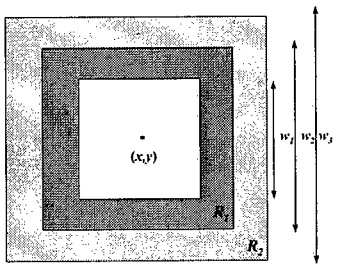


Fig. 1. Configuration of the surrounding regions on the current pixel (x,y).

3. EXPERIMENTAL RESULTS

3.1. ROI Selection

For comparison study of the texture analysis methods, 172 ROIs, with each ROI having 128 × 128 pixels, were selected from our database of digitized mammograms, which were digitized with a Lumisys laser film scanner with a pixel size of 100 × 100 μm^2 and 12 bits per pixel. Among the selected 172 ROIs, 72 ROIs were positive, containing the clustered microcalcifications, and 100 ROIs were negative, containing only normal tissues. Positive ROIs included clustered microcalcifications in dense regions and/or in glandular tissues. All of the clustered microcalcifications in positive

ROIs were verified by an expert mammographer based on visual criteria and biopsy results. The clustered microcalcifications are defined as containing three or more microcalcifications within an ROI (i.e., 1.28×1.28 cm^2). Negative ROIs include various breast areas involving ducts, breast boundaries, Cooper's ligaments, blood vessels, film artifacts, a single large calcification with benign characteristics, and/or glandular tissues.

3.2. Conventional Texture Analysis Methods

Three conventional texture analysis methods were evaluated with the same database, and all the textural features were obtained from 12 bits gray-level images.

3.2.1. Spatial Gray-Level Dependence Method (SGLDM)

The SGLDM [6] is based on the probability, $p(i,j|d,\theta)$, that two pixels, which are located with intersample spacing distance d and angle θ, have gray level i and gray level j. The thirteen textural features [6][10] are measured from the probability matrix, which are *energy, entropy, correlation, local homogeneity, inertia, sum average, sum variance, sum entropy, difference average, difference variance, difference entropy, and information measure of correlation1,2*. In this study, we computed four spatial gray-level dependence matrices according to four different directions (θ = 0°, 45°, 90°, and 135°) with a given distance d, and calculated textural features for each matrix.

3.2.2. Gray-Level Run Length Method (GLRLM)

The GLRLM [7] is based on the number of times, $g(i,j|\theta)$, that the picture contains run length j of gray level i in the given direction θ. Four gray level run length matrices are computed according to four different directions (θ = 0°, 45°, 90°, and 135°). The five textural features [7] are measured from each matrix, which are *short runs emphasis, long runs emphasis, gray level nonuniformity, run length nonuniformity, and run percentage.*

3.2.3. Gray-Level Difference Method (GLDM)

The GLDM [8] is based on the probability of occurrence that two pixels separated by a specific displacement vector δ have a given difference. In this analysis, four kinds of displacement vectors are considered, such as (0, d), (-d, d), (d, 0), (-d, -d), where d is intersample spacing distance. The five textural features [8] used in the experiments are *contrast, angular second moment, entropy, mean, and inverse difference moment.* In this study, we computed the probability density functions according to four kinds of displacement vectors and calculated textural features for each probability density function.

3.3. Classifier

The classification algorithm used in this paper is a three-layer backpropagation neural network [11]. A nonlinear sigmoid function with "0" and "1" saturation values is used as the activation function for each neuron. In the training process, the weights between the neurons are adjusted iteratively so that the difference between the output values and the target values is minimized. The weight values are updated by iteration as follows:

$$w_{ji}(l+1) = w_{ji}(l) + \eta\delta_j o_i + \mu\left[w_{ji}(l) - w_{ji}(l-1)\right] \tag{5}$$

where w_{ji} is the weight value from the ith to jth neurons, o_i is the ith element of the actual output pattern produced by an input pattern, η is the learning rate, l is the number of epochs, δ_j is the error signal, and μ is a momentum parameter. In this study, the learning rate η and the momentum μ are 0.08 and 0.7, respectively. To evaluate the network performance during the learning process, a global error measure is given as

$$\varepsilon_{RMS} = \sqrt{\frac{\sum_{g=1}^{G}\left[o_g - t_g\right]^2}{G}} \tag{6}$$

where o_g and t_g are the output value and the target value of neural network for the gth input pattern, respectively, and G is the number of training patterns. In this study, the learning process is stopped when the RMS error, ε_{RMS}, is less than 0.1.

3.4. Classification Results

To study on the efficacies of pattern classification by the jack-knife method, the 172 ROIs were partitioned arbitrarily into training and test sets, i.e., each set consists of 86 ROIs containing 50 negative ROIs and 36 positive ROIs. All the textural features were normalized by sample mean and standard deviation of training set. The LABROC1 algorithm developed by Metz et al. [12] was used to fit the outputs of the neural network obtained by the test set. The area under the ROC curve, A_z, is used as a measure of the classification performance. Optimum number of hidden neurons was analyzed for better A_z. Also, optimum distance d for the SGLDM and the GLDM, and optimum threshold q for the SRDM are analyzed. Figure 2 denotes comparisons of the classification performances for texture analysis methods by means of the ROC analysis. Figure 3(a) shows the comparison of four ROC curves at the optimal performance of each method performed by the jack-knife method.

We also studied the classification performance of texture analysis methods by using the round-robin method. When there are *D* sample patterns, this procedure trains the classifier with *D-1* samples, then uses the one remaining sample as a test sample. Classification is continued in this manner until all *D* samples have been used once as a test sample. Figure 3(b) shows the comparative result of the classification performances by the round-robin method in terms of the ROC analysis.

TABLE 1 shows the computation time required to extract features from an ROI, with a 128 × 128 pixels and 12 bits per pixel. All programs were written in C language and executed on a HP workstation (715, 100 MHz). The SGLDM was very time-consuming on 12-bit processing. From the viewpoint of classification accuracy and computational complexity, it is apparent that the SRDM is superior to the other methods.

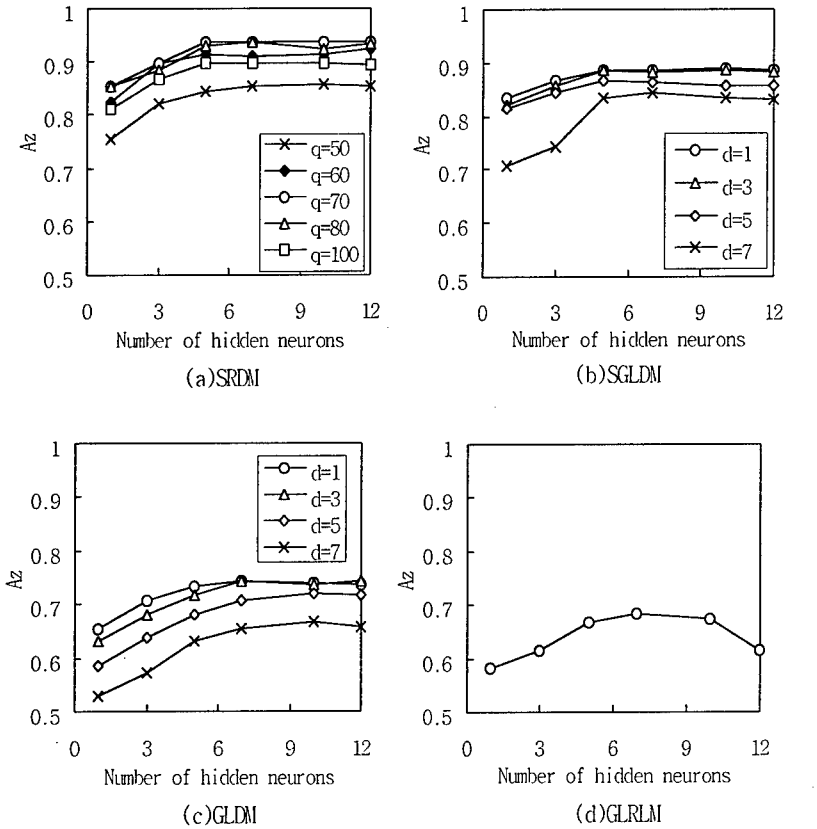


Fig. 2. Comparisons of the classification performances for texture analysis methods.

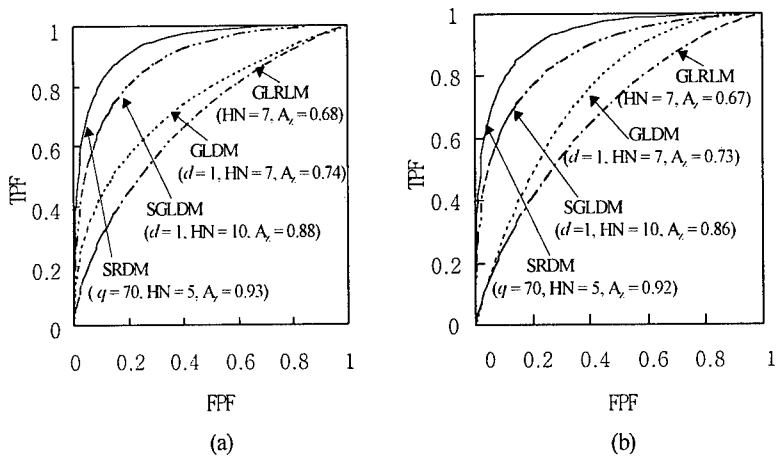


Fig. 3. (a) The comparison of ROC curves at the optimal performance of each method performed by the jack-knife method. (b) The comparison of ROC curves of each method performed by the round-robin method. Here TPF and FPF are the true-positive fraction and false-positive fraction, respectively, and HN denotes the number of hidden neurons.

TABLE 1. The comparisons of time required to extract features from an ROI, with a 128 × 128 pixels and 12 bits per pixel in HP workstation (715, 100 MHz).

Texture Analysis Methods	Time (seconds)
SRDM	0.65
SGLDM	681.6
GLRLM	8.75
GLDM	0.3

4. CONCLUSIONS

The goal of this work was to find the most useful texture analysis method performed in the spatial domain for the detection of clustered microcalcifications on mammograms. We performed comparative studies of the performances between the SRDM and the three conventional texture analysis methods. To evaluate the classification performances, the ROC analysis was performed. In spite of the limited number of cases, the performances of the SRDM are very promising. Further investigation of the effectiveness of the SRDM will be conducted with a large database in order to evaluate the SRDM for real clinical use in detecting the clustered microcalcifications on mammograms.

References

[1] D. B. Kopans, **Breast Imaging**, Philadelphia: J. B. Lippincoff company, 1989, ch. 5, pp. 81-95.

[2] H. P. Chan, K. Doi, S. Galhotra, C. J. Vyborny, H. Macmahon, and P. M. Jokich, "Image feature analysis and computer-aided diagnosis in digital radiography. 1. Automated detection of microcalcifications in mammography," **Med. Phys.**, vol. 14, no. 4, pp. 538-548, Jul/Aug 1987.

[3] Y. Wu, K. Doi, M. L. Giger, and R. M. Nishikawa, "Computerized detection of clustered microcalcifications in digital mammograms: Application of artificial neural networks," **Med. Phys.**, vol. 19, no. 3, pp. 555-560, May/June 1992.

[4] B. W. Fam, S. L. Olson, P. F. Winter, and F. J. Scholz, "Algorithm for the detection of fine clustered calcifications on film mammograms," **Radiology** , vol. 169, no. 2, pp. 333-337, November 1988.

[5] J. K. Kim, J. M. Park, K. S. Song, and H. W. Park, "Detection of clustered microcalcifications on mammograms using surrounding region dependence method and artificial neural network," in press on **Journal of VLSI Signal Processing**, 1997.

[6] R. M. Haralick, K. Shanmugan, and I. Dinstein, "Textural Features for Image Classification," **IEEE Trans. System, Man, and Cybernetics**, vol. SMC-3, no. 6, pp. 610-621, November 1973.

[7] M. M. Galloway, "Texture Analysis using Gray Level Run Lengths," **Computer Graphics and Imag. Processing**, vol. 4, pp. 172-179, 1975.

[8] J. S. Weszka, C. R. Dyer, and A. Rosenfeld, "A comparative study of texture measures for terrain classification," **IEEE Trans. Syst., Man, Cybern.**, vol. SMC-6, pp. 269-285, 1976.

[9] C. E. Metz, "ROC Methodology in Radiologic Imaging," **Investigative Radiology**, vol. 21, no. 9, pp. 720-733, September 1986.

[10] H. P. Chang, B. Sahiner, N. Petrick, K. L. Lam, and M. A. Helvie, "Effects of pixel size on classification of microcalcifications on digitized mammograms," **Proceeding of the SPIE**, vol. 2710, 1996, pp. 30-41.

[11] L. Fausett, **Fundamentals of neural networks**, Englewood Cliffs, NJ: Prentice-Hall, 1994, ch. 6, pp. 289-333.

[12] C. E. Metz, J. H. Shen, and B. A. Herman, "New methods for estimating a binormal ROC curve from continuously-distributed test results," presented at the 1990 Annual Meeting of the American Statistical Association, Anaheim, CA, August 7, 1990.

Neural Nets in Boundary Tracing Tasks

Stewart Crawford-Hines
Department of Computer Science
Colorado State University, Fort Collins, CO 80523 USA
sgcraw@cs.colostate.edu
http://www.cs.colostate.edu/~sgcraw

Charles W. Anderson
Department of Computer Science
Colorado State University, Fort Collins, CO 80523 USA
anderson@cs.colostate.edu
http://www.cs.colostate.edu/~anderson

Abstract

Our focus is to use neural networks to interactively assist in the initial segmentation of medical imagery, through learning the characteristics of a contour being traced and projecting ahead a trace whose initial few pixels were specified. To date, much of this work is done manually, since automatic techniques have yielded less than satisfactory results due to prerequisite background knowledge and noise in the data. In our framework, the expert interacts with the network to provide the context, and the network learns the characteristics of the (potentially noisy) locality and continues the task until further guidance is needed. We present here an initial application of this approach to brain MRI's, and we discuss our initial evaluation of neurologically-inspired preprocessing on the input pixel space. Our research directions are discussed.

1 INTRODUCTION

We are focusing on providing real-time learning and trace-ahead capabilities for region definition in image analysis tasks. In current medical image analysis, the reference standard for region delineation is an expert's manual outlining of the region. In certain domains, such as tumor identification, automatic delineation has made some modest success (for example [7]). However, as Johnson, et.al. [4], note: *"Although image segmentation and contour/edge detections have been in-*

0-7803-4256-9/97/$10.00

vestigated for quite a long time, there is still no algorithm that can automatically find region boundaries perfectly from clinically obtained medical images. There are two reasons for this. One is that most of the image segmentation algorithms are still noise sensitive. The second reason is that most segmentation tasks require certain background knowledge about the region(s) of interest."

Our model here is that a human expert sets down the initial several pixels of an image boundary, and a neural network continues the task by learning the local landscape and continuing through similar image territory as originally identified. One characteristic of neural nets is an adaptability to noise, and thus if the initial image territory is noisy, the network could learn to navigate through it, addressing the first concern above. In addressing the second concern, we note that hole-scene analysis, a straightforward task for a human expert, has proved exceedingly difficult to automate. The expert/network combination we set forward capitalizes on what each does best: the expert to provide global perspective and context, and the network to quickly analyze and work through similar local neighborhoods.

We have focused on neural networks as the learning mechanism due to their very general abilities. In earlier studies, we demonstrated their facility in learning non-linear region discriminations[1].

2 An MRI Application

Figures 1 and 2 show sagittal sections of MRI data from the National Institute for Mental Health (NIMH). This data is from an ongoing morphometric analyses at NIMH of the brains in monozygotic twin pairs[2]. Figure 1 shows the raw MRI image. Their initial image processing task is to subtract off the non-cerebral material in this image, resulting in Figure 2. They currently do this manually with a simple pixel eraser tool.

Figure 1: Raw data: sagittal MRI section

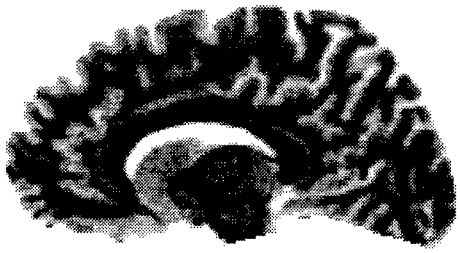


Figure 2: Cerebrum only, cleaned manually. (Orientation and contrast levels have been standardized, so this will not match exactly to the raw original.)

3 Neural Networks for Boundary Tracing

A model of our interaction scenario is illustrated on an enlarged set of pixels, shown in Figure 3. The darkest pixels represent a trace of 120 pixels. The first few pixels on the left were traced manually with a cursor over the image; that pixel segment became the neural nets exemplar to learn its contour, and then project the trace along the contour learned.

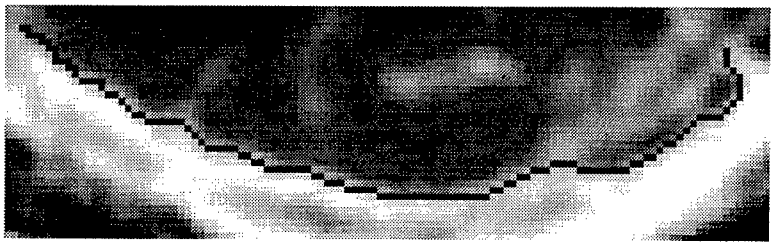


Figure 3: Enlargement: Network-traced path through the grey-scale landscape.

The contour is learned and followed by tracking characteristics of pixels to the left, to the right, and on the directed path, as in Figure 4. The neural net evaluates possible next pixels on the contour, based on what it has been trained on in the past.

3.1 Neural Net Design Issues - Output Representations

Our initial neural network design had one output unit, providing a single value on the range [0,1]: a low evaluation indicates a pixel is off to the left of the contour, a high evaluation indicates off to the right, and a value near 0.5 indicates the pixel is on the desired contour. The network learns an evaluation function that produces a

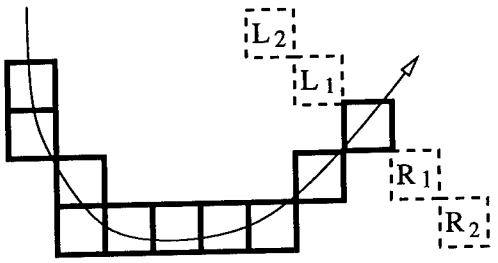


Figure 4: A path and its neighbors

smoothly changing value as a pixel and its neighbors change from left-of-contour values, to on-contour values, and then to right-of-contour values.

An alternative network design we studied also has one output unit, but this unit produces a low value for pixels centered on the contour, and a high value for off-contour pixels. This style of output is a feature-detector unit, where the output unit goes low on recognizing the contour, and stays high in non-contour regions.

3.2 Neural Net Design Issues - Training

Our initial experiments demonstrated the smooth-evaluation-function output unit works well when following a gradient, or ramp edge (for example, see Figures 3 and 5). Unfortunately, this is unworkable if the network is trying to learn to follow a thin line rather than a gradient. When following a line, the local neighborhoods off to the left and right side of the line are similar, and since they are expected to produce different outputs, this is no longer a functional form and thus can't be learned.

Exemplars of the contour for training are easy to derive, given an established contour in the image. Over the training set, the true extension of the curve for several pixels ahead is known, and can be added to the training set. A key issue, though, is the generation and spacing of negative exemplars. The set of possible extensions considered for each point needs to be looked at in the known training set, and appropriate non-contour training values established.

3.3 Neural Net Design Issues - Input Representations

There are a variety of options for representing the input pixel space:

- raw pixel values
- filtered inputs (Laplacian, Sobel, ...)

- masks based on neurologically-inspired models (center-surround, directed gradient, ...)

Since neural nets can automatically extract high-order moments from the data, it may seem best to just feed raw neighborhood pixel data into the network, and let it automatically learn its best model. This is the strategy used by the path-following system ALVINN [3].

However, in our application, efficient learning is also an issue, since one goal of our systems is to keep pace with human operators. Appropriate preprocessing of inputs should be able to accelerate the contour learning.

4 Early Results

Figure 5 illustrates several network-generated traces separating the cerebrum from its surrounding tissue. Each was generated in a clockwise direction. The network used for this result had a smooth-evaluation-function output and normalized pixel value inputs.

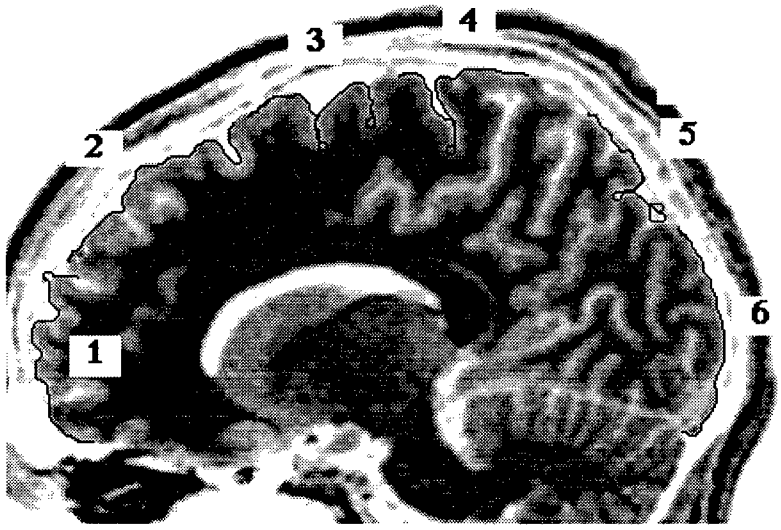


Figure 5: Net traced segments. (Note: this has been contrast enhanced and lightened to better show the traces; the original is a color trace on a full-range grey-scale image).

The initial 25 pixels of trace 3 were used as the training set. The network continued tracing ahead until it ran into trouble, in areas of the contour unrepresented

by the training set. The other traces represent restarts of the network-generated tracing (without further training) in new areas of the image. The end of trace 5 shows an area where the brain boundary is indistinct, and confounded by possible ghost structures from the MRI; the network performance is degraded by the region's similarity.

5 Experimental Evaluation of Input Representations

We performed some initial experiments aimed at verifying the hypothesis that preprocessing of inputs speeds learning. The experimental design used 3 inputs spaces $\times$ 2 outputs $\times$ 2 tasks. The speed of learning was quantified by the time taken to reach an RMS error of 0.1 from the target values

The three input representations consisted of the raw pixel values and two neurologically inspired models, illustrated here in Figure 6. The *center-surround* filter

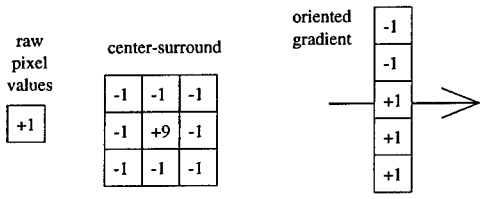


Figure 6: Input filters.

is based on early retinal processing, while the *oriented gradient* represents an oriented filter, such as those in the complex cells. These three filters evaluate to +1 when over a constant area.

The two output spaces were those discussed above: the SEV (smooth-evaluation-function), and the FD (feature-detector).

The two tasks were 1) edge following and 2) line following. A basic graphic was generated to minimize confounding influences of noise and texture in real images, shown here in Figure 7.

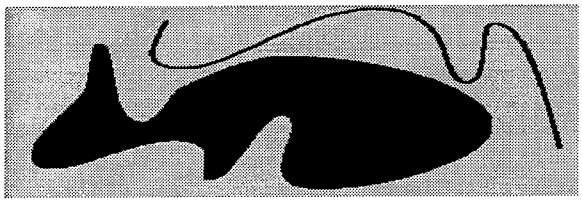


Figure 7: Line and Gradient test figures.

The neural network topology consisted of one input layer with five inputs, preprocessed as discussed above; there were 10 hidden units and one output unit (either SEV or FD). Other parameters held constant across the runs were: the initial training segment used; the pseudo-random network initialization; and the learning and momentum rates.

5.1 Basic Results

The following table summarizes the full set if initial results:

	SEV - Edge Task	SEV - Line Task	FD - Edge Task	FD - Line Task
Raw Pixels	traces well; learns very fast	X	traces well	traces ok
Center-Surround	traces well; learns fast	X	traces well; learns fast	traces well
Oriented Gradient	traces well	X	traces well; learns fast	traces well; learns fastest

Table 1: Summary of Initial Results

In one specific case, the SEV output with raw pixel inputs on the edge-following task was the fastest learning system, reaching its learning objective in 200 epochs. With the other two input representations, several thousand training epochs were required. The drawback of the SEV output model is that it works **only** on edges, however, and fails miserably when learning to follow a line.

With the FD output model, both tasks could be learned well. For the edge-following task, the center-surround and oriented-gradient filters both reached their learning criteria in 100 epochs vs. 800 epochs for raw pixel inputs. Equivalent improvements were evident for the line-following task.

It is interesting to note that the SEV outputs with raw pixel inputs was the fastest learning, but most specific combination tested. In a way, they are custom fit for each other. On an edge, the pixels on one side will all have low values and high values on the other; it should be easy to learn a smooth mapping onto [0,1] in this case.

Using the FD output model, the network could learn both tasks, and in this case the preprocessed inputs yielded faster learning. This neurologically-inspired com-

bination was a more general learning mechanism, even though somewhat slower in learning overall.

6 Research Directions

We are continuing to study various architectures appropriate for the task. An additional benefit from the feature-detector style output is a quantification of a "confusion" measure (i.e. output neither high nor low), for when the network needs to fall back to the human expert for intervention. The ability to know when the network is outside of its domain of expertise is a key implementation detail when adding such automated assistance onto existing tracing tools.

The traces of Figure 5 were generated using a very local 5-pixel neighborhood in searching ahead. This works better in a relatively noise-free domain, such as the high-resolution photography. However, when noise can interrupt a contour, the network must have a larger perspective to continue past the noise. Some other filters we plan to try are line-extension fields, analogous to those recently identified in complex cells.

Additionally, there are many basic engineering decisions in selecting and/or appropriately weighting training data, when many neighborhoods along the contour are redundant, and a few key cases capture the essence of the contour in its overall environment.

This tracing model shows promise. On straightforward contours, with 20 initial pixels of training data, contours can be followed continuously for hundreds of pixels. What remains to be measured is how well this squares with a ground-truth of an expert's delineation, and over how complex a landscape a network can be adequately trained.

A further extension of this work is into contour identification across the 3D volume composed of many parallel slices. We plan to explore contour extension on adjacent layers, without further training. And when slices are sufficiently well registered, several traced layers could analogously be used to propagate the contour identification to succeeding layers.

References

[1] S.Crawford-Hines & C.Anderson, "Interactive Region Bounding with Neural Nets", *Proceedings of the International Workshop on Neural Nets Applied to Control & Image Processing*, November 1994, pp. 58-61

[2] Hyde, Stacey, Coppola, Handel, Rickler, & Weinberger, "Cerebral morphometric abnormalities in Tourette's syndrome: a quantitative MRI study of monozygotic twins", *Neurology*, 1995; 45:1176-82.

[3] T.M.Jochem, D.A.Pomerleau, & C.E.Thorpe, "Vision-Based Neural Network Road and Intersection Detection and Traversal," *IEEE Conference on Intelligent Robots and Systems*, August 5-9, 1995.

[4] C.Johnson, R.MacLeod, & J.Schmidt, "Software Tools for Modeling, Computation, and Visualization in Medicine", *CompMed 94 Proceedings*, World Scientific, 1995.

[5] J.Malik & P.Perona, "Finding Boundaries in Images", *Neural Networks for Perception*, Academic Press, 1992, pp.315-344.

[6] M.Özkan, B.Dawant, & R.Maciunas, "Neural-Network-Based Segmentation of Multi-Modal Medical Images: A Comparative and Prospective Study", *IEEE Transactions on Medical Imaging*, v.12, #3, 1993, pp.534-554.

[7] Sai Raya, "Low-Level Segmentation of 3D Magnetic Resonance Brain Images - A Rule-Based System", *IEEE Transactions on Medical Imaging*, v.9, #3, 1990, pp.327-337.

NEUROCOMPUTING APPLICATIONS IN POST-OPERATIVE LIVER TRANSPLANT MONITORING.

D.G. Melvin[1], M. Niranjan[1], R.W. Prager[1], A.K. Trull[2], V.F. Hughes[2].
[1]Department of Engineering, University of Cambridge, UK.
[2]Clinical Biochemistry, Addenbrooke's NHS Trust, Cambridge, UK.

ABSTRACT

This paper explores the potential for the application of neurocomputing technology to the domain of post-operative liver transplant monitoring. The investigation compares a neural network model with two classical statistical techniques using biochemical information obtained from a set of liver transplant patients. Each approach combines the results of a number of liver function tests to predict the presence of allograft rejection. Each system is assessed, relative to the clinical gold standard, in terms of its overall accuracy and degree of advance warning offered. Applying non-linear methods does offer an advantage over the traditional linear techniques. The underlying structure of the data set has also been determined using k-means cluster analysis. This analysis suggests important directions for future investigation including the use of temporal information. Preliminary results of incorporating this temporal information are also presented.

INTRODUCTION

Transplantation of the human liver is currently the only viable therapeutic technique that can be applied to patients suffering from end stage liver failure. This procedure, while having advanced a great deal since its inception in the early 1980s, still has a great many risks associated with it.

Liver transplant patients must be maintained on immunosuppressive drug therapy. This deliberately inhibits their immune system from detecting and destroying their new liver.

Substantial advances in the transplantation arena, to date, have involved the improved understanding of the graft rejection process and the development of more specific immunosuppressive drugs. While the improvements in the available drugs have allowed better control of rejection, the drugs are still far from perfect. In particular these drugs are still among the most toxic prescribed to any population of patients.

Liver transplant patients are especially difficult to manage since their absorption / metabolism of the immunosuppressive drugs is affected by vari-

0-7803-4256-9/97/$10.00

ations in the function of their new liver [1, 2]. This unpredictability leads to difficulties in selecting the optimum dose of drug required to protect the patient from rejection *and* minimise the risk from adverse side effects associated with over immunosuppression. This requires the clinical staff to monitor the patient frequently and make adjustments to their drug regime based on information gleaned from a number of biochemical and haematological tests. These tests include liver function tests (LFTs) and blood cell type/count analyses designed to indicate whether the patients immune system is invoking some response (either fighting infection or rejecting the new liver). More subjective, clinical, indicators include the general condition of the patient and their speed of recovery from surgery.

Many of the modifications to the drug therapy are made in response to suspected or confirmed rejection of the liver. The presence of a rejection process may not be obvious from the available biochemical data until the rejection episode is relatively well advanced. Thus, dose changes may be relatively late and considerable cellular damage may already have occurred.

The risk of rejection is at its greatest in the initial period following transplantation. At this early stage, the clinical staff are concerned primarily with preventing graft rejection and they maintain the patient on a higher level of immunosuppression. Consequently, adverse side effects are most common at this time. This initial concern with the avoidance of rejection has motivated our research to focus on predicting impending rejection at an earlier stage. This task forms an integral part of the final goal of assisting with the determination, day to day, of the appropriate dose of immunosuppression.

The primary objective of this study has been to explore the feasibility of using neurocomputing to assist with post-operative monitoring of liver transplant patients. Applications include categorising the types of risk that can affect these patients and using dynamically acquired and historical data to assess the impact that such risks will have on their short and long term management.

Other researchers [3] have explored the use of connectionist techniques in this domain but have used the technology in a static mode which predicts long term graft survival based on pre-operative and early post-operative data. In our application, post-operative, on-line, data is analysed as and when it becomes available. This research explores the niche for increasing the degree of advance warning of impending rejection available to the clinical staff so that pre-emptive modifications to the immunosuppressive drug therapy may be made.

In this paper we compare three different classification systems in the performance of this type of task: logistic regression (LR), linear discriminant analysis (LDA) and multi layer perceptrons (MLP). The objective has been to determine the degree of advantage, relative to the well known statistical techniques, offered by the neurocomputing approach.

Each of these systems were designed to take the LFT results as input and yield, as output, a measure of the risk of rejection. The performance of each system was evaluated in terms of two criteria:

Biochemical Test Information				
Test Name	**Class**	**Ref. Range**	**Units**	**Half Life**
GST	Liver Enzyme	≤ 10	$\mu g L^{-1}$	$\leq$ 1 hour
ALP	Liver Enzyme	30-135	UL^{-1}	40 hours
ALT	Liver Enzyme	0-50	UL^{-1}	47 hours
BILI	Liver Excretion	0-17	$\mu mol L^{-1}$	Variable

Table 1: Selected Biochemical Test Information.

- Predictive accuracy for rejection.
- Ability to offer earlier warning.

The techniques used to perform these evaluations on the systems are described in the later sections of this paper.

AVAILABLE DATA SOURCES

A database was constructed using information collected by following the test results of 95 liver transplants in a total of 80 patients (15 re-transplants occurred). Their biochemical and haematological status was collected for up to 100 days following transplantation.

This paper focuses on a specific subset of this database. A more detailed account of the full database structure and content together with the results of a number of other analysis techniques is available in [4].

The database was refined to focus on a few, frequently used, LFTs specifically GST[1], ALP[1], ALT[1], BILI[2]. The numeric values for each of these tests were extracted to form a number of 4 dimensional feature vectors (or frames) corresponding to the measurements on a patients on a given day. Each frame was augmented with a tag which indicated if the data originated from a patient who was undergoing a rejection[3] or not. This refinement process yielded 3206 data frames (56 reject, 3150 non-reject). Figure 1 shows an example of the profile followed by these four LFTs over the course of a single patients post-operative period. Tables 1 and 2 outline the general biochemical characteristics of these four tests. A detailed description of the mode of operation of these tests is available in [4]. In summary, these tests detect liver damage by measuring the amount of enzyme released from damaged cells (liver enzyme) and the degree of blockage of the liver bile drainage system (liver excretion). The range of values, exhibited by a population of healthy volunteers, is shown in Table 1 as the reference range. The range exhibited by our liver transplant patients is shown in Table 2. The distribution of the

[1] Serum concentration of α-Glutathione S-transferase (GST). Serum Alkaline Phosphatase (ALP) and Alanine Transiminase (ALT) activities.

[2] Serum bilirubin concentration (BILI).

[3] In line with current clinical practice, only biopsy confirmed rejections were used to identify rejection.

Biochemical Test Statistics				
	ALP	**ALT**	**GST**	**BILI**
Reject				
max	1720	1940	1140	479
min	63	44	3.9	17
median	282	249.5	25.5	144.5
Non-Reject				
max	3890	9440	47250	993
min	23	4	0.1	2
median	224	112	10	48

Table 2: Selected Statistics of the Biochemical Tests.

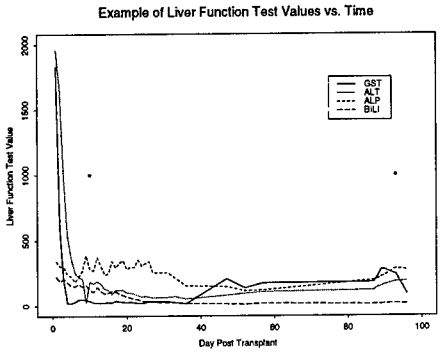


Figure 1: Example profile of LFTs over time. Two rejections occur in this example denoted by the dots (at y=1000).

test values is also known to be log-normal [4] and consequently the data has been transformed logarithmically to account for this.

The frame set, described above, was segmented by a random selection process. This procedure chose a number of frames from each of the two classes (reject/non-reject) to create a training set. A random 25% portion of the data was excluded for use in testing. This procedure was repeated to yield a 100 pseudo-random training and test data sets for use in constructing/evaluating the performance of each of the classification systems.

MODEL DESCRIPTIONS

The task of interest here can be viewed as a multivariate two class prediction problem. The independent variables being the selected liver function tests (as provided by the data frames described above) and the dependent variable the risk of rejection in the immediate future. All of the models documented here apply this as the high level strategy with the individual solutions varying only in their implementation approach.

The sensitivity of each model, to the data used in its construction, has

been assessed by computing the distribution of the results obtained from the model over the 100 training/test data sets.

The regression parameters for the logistic regression model were computed on the training segments of the 100 data sets and analysed using the corresponding test segments.

The linear discriminant analysis model was based around a Fisher transformation [5, 6] of the 4D data space into a single discriminant dimension. Using this discriminant projection it was possible compute a single value for any 4D data vector. Defining a threshold value for this parameter allows a classifier to be constructed which reports rejection if the fixed decision threshold is exceeded.

The architecture used for the multi-layer perceptron approach was a fully connected feed forward network with four input units, five hidden units and a single output unit. The hidden and output units have a logistic transfer function. A small number of hidden nodes were chosen to mediate over-fitting of the relatively small data set. The network was trained on a balanced version of the data set having the "reject" class artificially expanded by duplication of the existing members of this class. The data was pre-processed to give the data zero mean and unit variance. The parameters used to perform this preprocessing were calculated on the training data but were stored to allow transformation of unseen/test data.

RESULTS AND ANALYSIS

Analysis Techniques

This section makes considerable use of the ROC [7, 8] analysis technique. This technique is frequently used to compare the performance of two class decision systems. In particular this paper makes use of the technique to directly compare the capabilities of the various classifiers applied to the liver transplant data. More specifically both the shape and area [9] under the ROC curve are used to show the differences in performance of the liver transplant classification systems. The ROC curves show both the strength of the system at detecting the rejections (sensitivity) as well as the degree to which the system is specific to rejection and tolerant of other phenomena (specificity). Each point on an ROC curve denotes these two quantities (expressed as a rate) for a chosen threshold value for the output of the classifier. Specifically the ROC curve is formed by selecting a series of threshold values for the output of the system (e.g. (output ≥ 0.5) $\rightarrow$ rejection) and evaluating how sensitive and specific the system is for a test data set. An optimal ROC curve is one where the test is highly specific (1 - specificity) = 0 and highly sensitive (sensitivity = 1) and leads to a "square" ROC curve which runs from (0,0) to (0,1) to (1,1). The line shown on the ROC curves, denoted "Reference", represents the result of a system which is random.

The area under the ROC curve can be used as direct comparison measure. It can be interpreted as the probability of correctly ordering a pair of examples

(one rejecting and one non-rejecting). Thus the closer the area under the ROC curve is to 1.0 the better the classification system is at performing its task.

The ROC curve and area for each of the models is given below. Specifically example ROC curves are supplied which show the performance of the model for a single data set. Alongside this, histograms of the area under the ROC are supplied. These show the distribution of the areas observed over the 100 training/test data sets. A similar histogram is supplied which depicts the area as a function of advance warning level.

Logistic Regression Models

Figure 2a depicts the general performance of the logistic regression based classifier. It can be seen that the classifier is noticeably better than random but that it is still far from optimal.

Figure 3a summarises the performance of this model over the available data sets and confirms that, in general, there is a approximately 75% chance of correctly ordering a random reject/non-reject pair (i.e. the classifier produces a larger output value for the reject example).

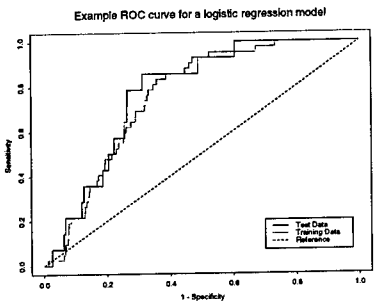
Figure 3b shows the performance decay, towards random, of the model as greater advance warning is expected. Specifically this shows the result of assessing the model on data obtained on days preceding the actual biopsy confirmed rejection. Figure 2b summarises this advance warning performance over the available random data sets.

The wide variation in the zero warning histogram can be attributed to the limited number of data points available in the testing set. Thus for a small change in the threshold value used in the classifier large changes can occur in the sensitivity and specificity *rates*. This phenomena combines with the variations induced by the different training sets to produce a considerably larger spread of areas under the ROC curve.

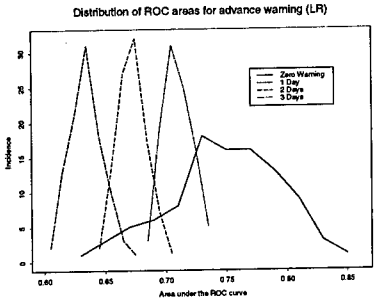
Discriminant Analysis Models

The result of sweeping the decision threshold through a range of values and computing an ROC for an example LDA model is shown in figure 2c. It can be seen that this system is also significantly better than a random classifier but less than optimal.

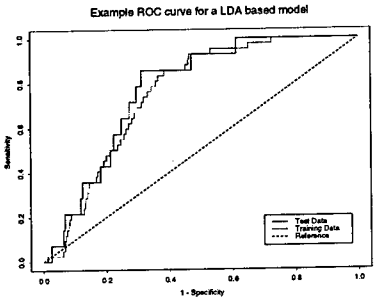
Figure 2d depicts the distribution of areas for the different data sets. It can be seen that there is a marginal improvement in performance obtained from the LDA model (relative to the LR model) with a slightly heavier right tail. This marginal improvement is also reflected in the advance warning performance. It can however be seen that there is a rapid degradation in the performance of the classifier as the degree of warning is increased (i.e. the area under the curve tends rapidly to 0.5). The speed with which it degrades is of considerable importance in terms of developing a system which can yield an acceptable level of advance warning.



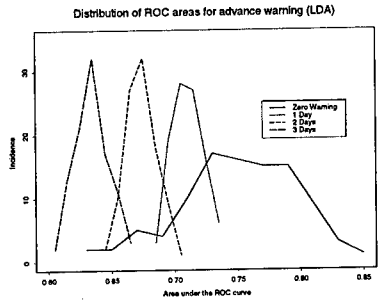
(a) Example logistic regression (LR) classifier ROC.



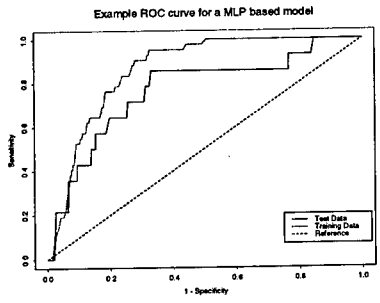
(b) Histograms of area under the ROC (LR model).



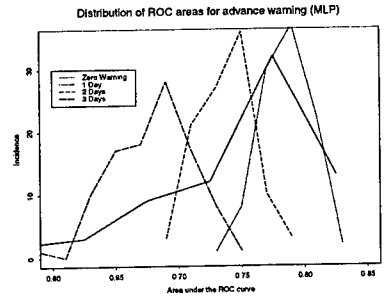
(c) Example linear discriminant analysis (LDA) classifier ROC.



(d) Histograms of area under the ROC (LDA model).

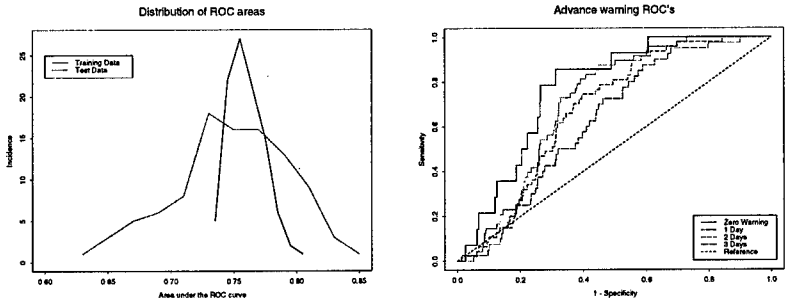


(e) Example multilayer perceptron (MLP) classifier ROC.



(f) Histograms of area under the ROC (MLP model).

Figure 2: Classifier performance expressed both as Receiver Operating Characteristic (ROC) curves and histograms of the area under the curve computed for the different cross validation sets. The histograms also depict the advance warning performance of each system.



(a) Distribution of the area under the ROC curve for different cross validation data sets (both test and training).

(b) Logistic regression model performance in terms of providing advance warning.

Figure 3: Generalisation and advance warning performance for the logistic regression model.

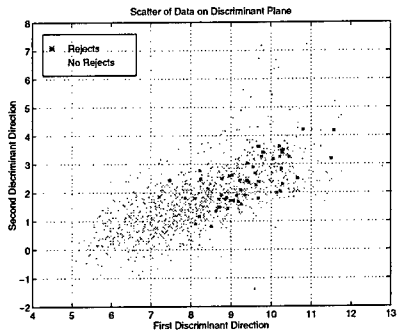
Multi-Layer Perceptron Models

The performance of a MLP system is shown in figure 2e. It can be seen that there is an enhanced performance relative to the more traditional LR and LDA models. However none of the systems investigated, including the MLP, show optimal performance. It can also be seen that the MLP model performs marginally better on the training set than on the test set. This implies that, at least with the example shown in figure 2e, there is a small degree of over fitting of the the data in the MLP model. Conversely figure 2f shows that the MLP still performs better when assessed with the cross validation approach.

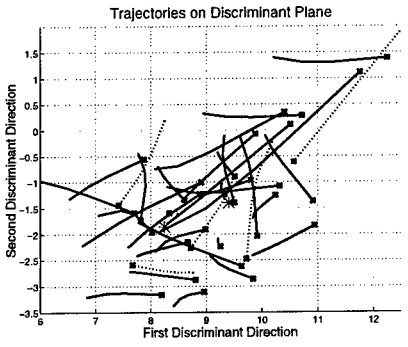
CONCLUSION AND FUTURE DIRECTIONS

This paper presented some preliminary results of applying neurocomputing technology to predicting allograft rejection in liver transplant recipients. Posing the problem as a classification task, it provided a comparison of classification performance (in terms of overall accuracy and degree of advance warning) of a MLP based system relative to traditional statistical approaches. The liver function tests, taken in combination, do seem to carry useful extra information capable of predicting rejection two or three days ahead in time. The non-linear neural network classifiers outperform the classical linear approaches for this task.

The non-optimal behaviour of these preliminary models can be partly attributed to the overlap in the distributions of the data. Considerable overlap between the reject and non-reject distributions can be observed when visualising the data in 1 or 2 dimensions (see Figure 4a). This overlap is attributable to variation in severity of rejection, to some degree of mis-labelling of the data



(a) Scatter Plot of the reject/non-reject data on the 2D Fisher plane.



(b) Trajectories across the Fisher plane.

Figure 4: Distribution of reject/non-reject data on the Fisher plane and the trajectory followed by data prior to rejection episodes.

Cluster Centres						
Cluster	**GST**	**ALT**	**ALP**	**BILIRUBIN**	**Nr**	**Nnr**
1	6.4	69.7	144.0	28.6	6	1705
2	25.9	234.5	392.3	132.4	60	1758

Table 3: Cluster centres for k=2 in original units. Nr and Nnr columns show the number of vectors in each cluster that are marked as rejecting and non-rejecting in the original labelling.

forming the non-reject class and inter-patient variability. K-means [10] clustering has been applied to this database in an effort to identify structure in the data and any potential mis-labelling. Table 3 shows the result of splitting the data into two clusters. The clinical reference ranges listed in Table 1 compare well with the values selected by the k-means algorithm for class 1. This class contains very few examples of measurements associated with rejection. Class 2, conversely, shows elevated values for the LFTs and is responsible for the majority of vectors associated with rejection. However this class also contains a substantial number of members which were originally labelled as non-rejecting. A sizeable proportion of these samples occur in the early post-operative period and are not associated with rejection but with damage caused during the transplant procedure. Including trend information allows these early measurements to be interpreted more appropriately since initially there is a decline in the test values consistent with normal clearance.

Figure 4b shows the trajectory followed by a number of the patients as they approach a rejection episode (only patients having 5 consecutive measurements leading up to rejection are displayed). The solid lines identify those examples where the patient followed a consistent trend towards the rejecting

population centre. The dotted lines show the small number of example where they did not.

Principled techniques for the inclusion of this trajectory information are currently under investigation together with techniques for estimating values for missing data.

REFERENCES

[1] G.J. Burckart *et.al.*, "Cyclosporine absorbtion following orthotopic liver transplantation.," *Journal of Clinical Pharmacology*, vol. 26, pp. 647, 1986.

[2] W.E. Evans, J.J. Schentag, and W.J. Jusko, Eds., *Applied Pharmacokinetics, Principals of Therapeutic Drug Monitoring*, pp. 28–1 – 28–40, Edwards Brothers, Ann Arbor, MI, U.S.A., 1992.

[3] H.R. Doyle *et.al.*, "Predicting outcomes after liver transplantation. a connectionist approach," *Annals of Surgery*, vol. 219, no. 4, pp. 408–415, 1994.

[4] D.G. Melvin, "A comparison of statistical and connectionist techniques for liver transplant monitoring.," Tech. Rep. CUED/F-INFENG/TR.282, University of Cambridge, Department of Engineering, Dec. 1996.

[5] R.A. Fisher, "The use of multiple measurements in taxonomic problems.," *Annals of Eugenics*, vol. 7, pp. 179 – 188, 1936.

[6] C.M. Bishop, *Neural Networks for Pattern Recognition*, Clarendon Press, Oxford, 1995.

[7] J.R. Beck and E.K. Shultz, "The use of relative operating characteristic (ROC) curves in test performance evaluation," *Arch. Pathol. Lab. Med.*, vol. 110, pp. 13–20, 1986.

[8] M.H. Zweig and G. Campbell, "Receiver-operating characteristics (ROC) plots: A fundamental evaluation," *Clinical Chemistry*, vol. 39, pp. 561–577, 1993.

[9] J.A. Hanley and B.J. McNeil, "The meaning and use of the area under a receiver operating characteristic (ROC) curve.," *Radiology*, vol. 143, pp. 29–36, 1982.

[10] J.A. Hartigan and M.A. Wong, "A k-means clustering algorithim," *Applied Statistics*, vol. 28, pp. 100–108, 1979.

CLASSIFICATION AND COMPRESSION OF ICEGS USING GAUSSIAN MIXTURE MODELS

Richard Coggins and Marwan Jabri
Systems Engineering and Design Automation Laboratory
Department of Electrical Engineering J03,
University of Sydney, 2006, Australia.
Email: richardc@sedal.usyd.edu.au, Fax: +61 2 9351 7209

Abstract

Implantable cardioverter defibrillators (ICD) administer high voltage shock therapies to terminate dangerous cardiac arrhythmias. Improving the functionality of these devices to include on-line diagnosis based on Intracardiac Electrogram (ICEG) morphology and to log dangerous signals is important for their more widespread use. It is essential that the ICD implement a signal compression scheme due to the limited memory in the device. We have fitted gaussian mixture models to the ICEG signals in order to investigate to what extent, non-linear data models are advantageous in this application compared to the traditional linear approaches used in the field and to explore the common features between classification and compression. Results of fitting the mixture models show that typically a single gaussian per class for classifiers and single gaussian prediction models for data compression are adequate data representations provided the data is preprocessed to remove non-stationary behaviour.

1 INTRODUCTION

This paper develops a framework for establishing performance bounds for ICEG classification and compression systems. The requirements for com-

0-7803-4256-9/97/$10.00

pression and classification are different in emphasis but share some common features. For the case of arrhythmia classification via morphology analysis (which improves diagnosis for some arrhythmia over systems which rely on heart rate alone), the states of the heart are being distinguished by trying to determine which type of signal the heart is producing given some previous knowledge about what types of signals are produced under both normal and abnormal conditions. For the case of ICEG data compression, a concise representation of the signal is sought, given knowledge of the signal in the past, but so as to retain a great deal of the detail in the signal. Thus, morphology classification is an extreme form of compression that retains only a labelling of the signal, whereas compression seeks to retain the diagnostically significant part of the local structure. In this paper we demonstrate the advantage of common scale and shift invariant preprocessing for these two applications and then using this representation, model the transformed signals with mixtures of gaussians. Gaussians are a good choice for real valued data as they render a model which may be interpreted in terms of well known parameters and may be easily manipulated to obtain relationships between variables. The fitting of a mixture of gaussians is described in [McLachlan and Basford, 1987].

Section 2 introduces the modelling steps and discusses in detail those steps common to both classification and compression. Section 3 describes the estimation procedure and bounds for ICEG morphology classifiers. The classifier bounds include both the case of separating NSR (normal heart rhythm) from VT 1:1 (dangerous rhythm) with examples of both available and the blind separation of NSR from VT given only examples of NSR. Section 4 describes the estimation procedure and bounds for ICEG data compression.

2 SOURCE MODELLING OF THE ICEG

The modelling procedure adopted is described by the following steps:

- Segmentation: The ICEG time series is segmented into "QRS complexes", which are 30 samples wide and contain the diagnostic information (see Figure 1).
- Identification: The non-stationary behaviour of the signal is removed and an initial model is selected.
- Estimation: The model is fitted using the Expectation Maximisation (EM) algorithm [Dempster et. al., 1977].

The model identification steps will now be described as these form the common preprocessing steps for both classification and compression. The description of the estimation steps is deferred to the respective sections on classification and compression.

The ICEG is a periodic non-stationary signal. There are several mechanisms for its non-stationary behaviour: the ICEG is a measure of the heart pumping blood in a normally synchronous manner which leads to periodic (seasonal) non-stationarity; the ICEG is subject to longer term influences such as daily cycles in metabolism, exercise, tissue growth, disease and aging. For

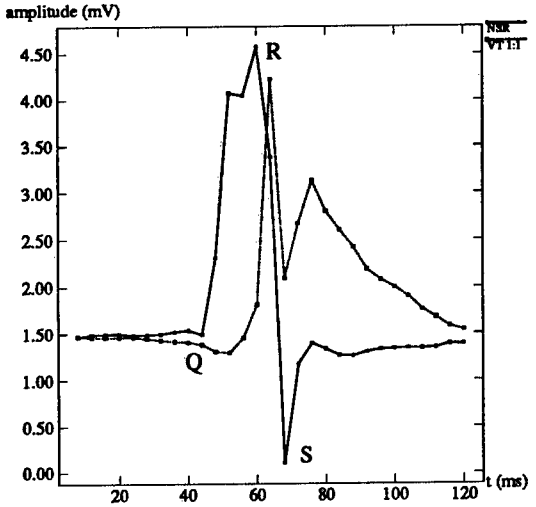


Figure 1: The morphology of NSR and VT retrograde 1:1. The letters QRS are conventional notation to label segments of the signal which correspond to those times when the heart is actually pumping blood. Hence, this period is often referred to as the QRS complex.

recordings[1] lasting from minutes to hours only the short term non-stationary behaviour is relevant. Due to segmentation, the variance in the period is reduced to that of the QRS complex detector which is about 3 sample intervals. The effect of variation in the R point detection (detection of the signal peak) can be removed by calculating the correlation of the current complex with the previous. The non-stationarity in the mean of the signal may be removed by the standard method of periodic differencing. Stationarity in the mean is indicated when the auto-correlation function quickly decays to small values after a few lags. The dashed line in Figure 2(c) shows the effect of periodic differencing on the ICEG. Figure 2(c) (solid line) shows evidence of a residual signal with a randomly varying amplitude. The origin of this residual non-stationarity is the A/D converter sampling the ICEG asynchronously resulting in "sample jitter". This jitter process can be described by the subsequent QRS complex segment $y(t)$ being identical to a complex $x(t)$ but with a random phase shift ϕ. Expressing $y(t)$ as a Taylor expansion of $x(t)$ to the first derivative term around t,

$$y(t) = x(t) + \phi x'(t) + \Theta \tag{1}$$

Thus, the residual non-stationary component in Figure 2(c) is identified as $x'(t)$ modulated by the random phase variable ϕ. The Taylor expansion to the first derivative term is an adequate approximation because $x(t)$ is

[1]Data in this paper is recorded in hospitals. Arrhythmia are induced artificially. Long recordings and natural arrhythmia are hard to get due to lack of field recording capability.

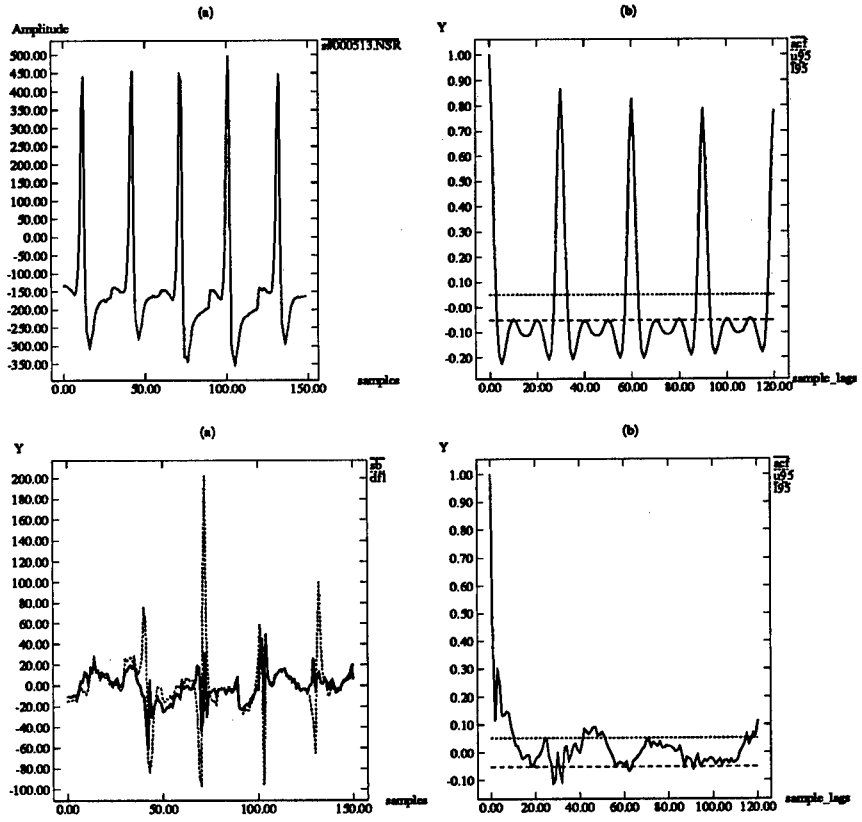


Figure 2: The effect of amplitude and phase locking to the previous beat. (a) The original segmented ICEG and (b) the autocorrelation function. (c) The residual ICEG after amplitude and phase locking (solid line) compared to the periodic differenced ICEG (dashed line) and (d) the resulting autocorrelation function. The horizontal lines are 95% bounds for a white noise sequence.

sampled well within the Nyquist criterion to avoid aliasing. The random phase ϕ is estimated by calculating the correlation of the periodically differenced sequence with the derivative of the previous beat. Hence, the jitter free seasonally differenced sequence is given by,

$$\Delta y(j) = y(j) - x(j) - \phi(x(j+1) - x(j-1))/2 \qquad (2)$$

This procedure may be enhanced by doing scaled periodic differencing instead of just periodic differencing in order to remove amplitude variations between cycles. This can be achieved by calculating a scaling factor A, from which the jitter free scaled seasonal difference,

$$\Delta y_s(j) = y(j) - Ax(j) - \phi A(x(j+1) - x(j-1))/2 \qquad (3)$$

is obtained. Figure 2(d) shows the effect of Equation 3 on the autocorrelation function and is used for compression. Similarly, Equation 2 can be used to

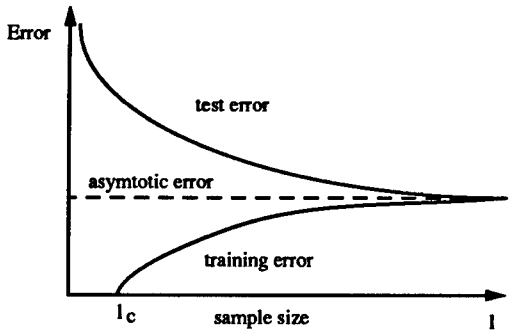


Figure 3: Learning curves for a classifier. E is the expectation of the classifier error over all possible training and test sets. l is the size of the data sample. As l becomes large the training and testing errors converge to a common value E_∞. Above a certain training set size l_c the training and testing curves can be modelled with power-law decays.

phase lock complexes for jitter free classification.

The remaining identification step is to determine the model order. For classification purposes the 30 dimensional phase locked complexes comprise the classifier inputs. For compression purposes, a choice about the size of the memory in the model needs to be made. The autocorrelation function after amplitude and phase locking to the previous complex shows that there is significant correlation in the first 4 lags. This suggests an autoregressive model for the time series with 4 AR coefficients corresponding to the 4 lags. A preliminary study on a small number of patients showed that a long term component in the model did not contribute significantly to a lower entropy estimate. Hence, short term model orders in the range 1 to 4 only were considered. This determines the vector for which the distribution is to be estimated as

$$\mathbf{x} = (x_n x_{n-1} \ldots x_{n-m}) \tag{4}$$

where $\mathbf{x}$ slides over the time series and m is the short term model order and M is the model capacity (number of basis functions used).

3 CLASSIFIER PERFORMANCE BOUNDS

3.1 Model Estimation for Classifiers

A classifier is determined by both its structure and the number of free parameters. Learning curves are defined as the expectation of the classifier error over all equally sized training and testing sets that may be randomly selected from the data sample of size l. In [Cortes et. al., 1994] it was shown that these learning curves can often be modelled by power-law decays to obtain an estimate of the asymptotic performance of the classifier. An illustration of these learning curves is depicted in Figure 3. In [Cortes et. al., 1995] it is demonstrated that these asymptotic estimates may then be used to bound

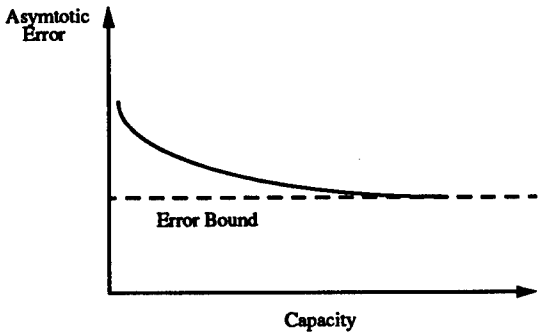


Figure 4: Example of how the asymptotic classifier error behaves as a function of classifier capacity. Eventually, a capacity is reached beyond which negligible improvement in classifier performance is achieved.

the performance of any classifier on that data by considering the asymtotic classifier error as the capacity is increased. An illustration of an asymptotic classifier error curve is shown in Figure 4.

3.2 Bounds for NSR and VT 1:1 Classification

The efficacy of a single decision classifier may be expressed in terms of the expected fraction of false negative and false positive detections. Here we use a classifier which derives the classification from the probability density model of the data. This more general representation provides a natural extension to a blind classification scheme which detects the presence or absence of NSR. A mixture of gaussians density model and the EM algorithm may be used to model labelled data using a procedure similar to that described in [Ghahramani and Jordan, 1994]. In order to fit the best possible model an evaluation of the appropriate number and form the of gaussians (capacity) per class is required. These issues could be evaluated using learning curves or other statistical tests [McLachlan and Basford, 1987]. As a starting point, a simple model was chosen, consisting of one spherically symmetric gaussian per class. Such a simple model will provide an upper bound for the classification error rate.

Table 1 summarises the misclassification bounds computed using the simple spherical gaussian per class. The % correct columns are averages over 10 different training and testing sets constructed by randomly splitting the sample sets in two and then refitting the density model. The sample complexes are sequentially phase locked to each other prior to fitting the models. The table shows that all patients indicate a misclassification rate of less than 1%. Since, the misclassification rates for the test sets are already very low, no attempt was made to obtain a tighter bound by considering the convergence properties of the training and testing error rates.

Patient	No. of Complexes		% Correct training		% Correct testing		% Correct blind testing	
	ST	VT	ST	VT	ST	VT	ST	VT
1	440	61	99.6	100	99.6	100	98.9	100
2	107	71	100	100	100	100	99.4	100
3	177	75	100	99.5	100	99.7	97.7	100
4	110	71	100	100	100	100	99.3	100
5	38	90	100	100	100	100	98.9	100

Table 1: Estimate of misclassification bounds for 5 patients with VT 1:1 retrograde conduction for both classification with training on the VT data and without training on VT (blind).

Bounds for a classifier which only has a measure of the NSR probability density may be computed by considering only the density estimation of the NSR data and assigning an arbitrary probability threshold for considering a data point to be a member of the NSR class. In this case the training set misclassification error is predetermined by the acceptable level of false positive detections. Testing the model on NSR only, then indicates the adequacy of the model as an estimate of the NSR complexes probability density at that error rate. By requiring the training error rate to be zero, the least probable training vector determines a probability threshold for NSR membership. The NSR testing set then gives an upper bound on the resultant false positive error rate. By then testing the model against VT with the same threshold the sensitivity to VT is determined. Table 1 shows that high classification performance is predicted with a single spherical gaussian modelling the NSR density. With a maximum of 2% false positive error rate the simple density model appears adequate and hence more complex models were not considered.

4 DATA COMPRESSION BOUNDS

This section describes the bounds calculated for the data compression of the QRS complex of the ICEG. Firstly, a generalisation of the model estimation procedure used for the classifier bounds is described which uses learning curves based on the estimated entropy of the data. The method is then applied to the ICEG data base and yields bounds for lossless data compression.

4.1 Model Estimation for Compression

The goal of estimation is to determine a probability distribution of the form, $P(\mathbf{x}) = \sum_{j=1}^{M} \pi_j G_j(\mathbf{x})$ where $\mathbf{x}$ is given by Equation 4 and G_j is the gaussian probability distribution function given by,

$$G_j(\mathbf{x}) = \frac{\exp\{-\frac{1}{2}(\mathbf{x}-\mu_j)C_j^{-1}(\mathbf{x}-\mu_j)^T\}}{(2\pi)^{m/2}|C_j|^{1/2}} \quad (5)$$

Figure 5 shows the procedure used to determine $P(\mathbf{x})$. When EM is used to fit the model it maximises the log probability that the model generated

```
Select a data sample and Identify an initial model;
Initialise sample-size;
For sample-size < max-samples; sample-size *= 2; {
    For n-splits = 0; n-splits < total-splits; n-splits++; {
        randomly split the sample in half
        into training and test sets;
        Use EM to fit model;
        Determine H_train(M), H_test(M);
    }
}
Estimate H_∞(M) and Increase M;
Continue until M determined;
```

Figure 5: Procedure for determining M and $P(\mathbf{x})$. The value of total-splits used was 10. The sample size was varied over a range of 8.

the training data, given by $L = \frac{1}{N}\prod_i^N \log_2 P(\mathbf{x}_i)$ where N is the size of the training set. Notice that L is closely related to the average information of the training data with respect to the model. In fact $L = -H(M,N)$ where H is an estimate of the entropy of the data source based on a sample size N and a model of capacity M. Hence, given a model and training and test sets both of size N, $H_{train}(M,N)$ and $H_{test}(M,N)$ can be calculated. Analogous to the procedure of [Cortes et. al., 1994] the behaviour of H_{test} and H_{train} is considered as N is increased[2]. In the limit as $N \rightarrow \infty$, $H_{test} = H_{train} = H_\infty$. H_∞ is then the source entropy estimate for the vector $\mathbf{x}$ as measured by the model $H(M)$. If both H_{test} and H_{train} approach this limit at equal rates then an unbiased estimate of H_∞ is obtained via

$$\hat{H_\infty} = \frac{H_{train}(M,N) + H_{test}(M,N)}{2} \tag{6}$$

If the rates of convergence are unequal the procedure of [Cortes et. al., 1994] can be used which involves fitting power laws to the learning curves for H over a range of N to deduce the unbiased estimate. Having obtained an estimate of H_∞, more complex models can be fitted by increasing M until no significant reduction in H_∞ is obtained. By this means, the capacity of the model from a finite sample is obtained, while also determining an upper bound on the entropy of the source.

4.2 Results for the ICEG

In this section mixture of gaussian models are fitted to the ICEGs of several patients who are candidates for ICD implantation using the procedures described in Section 2. Single patients and a single rhythm type for that patient are considered. Table 2 shows the variety of models required for various

[2]Actually $E[H]$ is calculated where the expectation is over all possible choices of training and testing sets. In practice only the expectation using a small number (10) of random choices is estimated.

Patient	Rhythm	M	N	H_∞	H_{lin}
s000418	NSR	3	688	8.6	9.3
s000418	VT	4	703	9.4	10.6
045.vts	NSR	3	719	7.3	9.0
045.vts	VTR	3	719	8.0	9.5
55.vts	NSR	3	719	6.9	7.9
55.vts	VTR	2	719	8.4	9.4
s000518	VF	4	719	10.1	10.3
s000533	NSR	2	719	5.6	5.9
s000533	VF	3	719	8.1	8.5
s000513	NSR	3	719	5.6	6.5
s000513	VT	2	719	6.1	6.5
s000513	VF	3	719	6.6	7.5

Table 2: ICEG probability model capacity and H_∞ estimates. The entropy estimates are in units of bits per (4ms) sample.

patients and various rhythms. The H_∞ shown is obtained using the simple estimate of Equation 6. The conditioned probability model $P(x|\mathbf{x}_s)$ can be easily derived from the fitted model to give the time series entropy estimate $H_\infty(x|\mathbf{x}_s)$. The column H_{lin} is included to compare the result with the fitting of a single gaussian. The model capacity M was determined conservatively by only taking higher values of M when there is no evidence for H_∞ increasing as determined by the bounding standard deviation curves. Having obtained the distribution of the data, the conditional entropy $H(x|\mathbf{x}_S)$ can be determined, where $\mathbf{x}_S$ are the previous short and long term outputs of the source. $H(x|\mathbf{x}_s)$ represents an upper bound on the true source entropy. The true source entropy will only be reached if the source is finite memory and autoregressive and it is stationary and ergodic. Of these conditions the ICEG is most likely to fail on stationarity, as it is difficult to guarantee that all higher order moments are stationary. Therefore, in practice only upper bounds on the source entropy of the ICEG data can be achieved.

The three heart rhythms NSR, VT and VF correspond to three therapeutic groupings of the ICD, being no therapy, pacing and defibrillation. The morphologies are usually noticeably different for the three rhythms. NSR and VT are quite periodic where as VF is quite variable in both period and amplitude. Table 2 shows that between 2 and 4 gaussians are sufficient to model the data. The reduction in H_∞ is both patient and rhythm dependent with between 2% and 20% reduction compared to the fit of a single gaussian. Over the entire data base of 146 patients a 15% reduction in the entropy estimate for the NSR class was obtained by fitting a mixture model, 11% for the VT class and 8.5% for the VF class.

5 CONCLUSION

Mixtures of gaussians models were chosen due to their clear interpretation and their utility for analysing both data compression and classification problems. Amplitude and phase locking was identified as an important common preprocessing step for both tasks. Classifiers were modelled using single gaussians per class and learning curves were used to determine misclassification bounds. Both the separation of NSR from VT 1:1 with training sets of both and blind separation were considered for 5 patients. Bounds close to 100% correct classification were indicated for both classification tasks. Data compression bounds were obtained using mixture of gaussians models fitted using the EM algorithm. The capacity of the models was determined by using a generalisation of the learning curve procedure using the entropy estimates. Experiments on the ICEG data showed that less than 5 gaussians were required for the modelling of the data and that the resulting reduction in entropy estimate over the single gaussian case was 8% to 15% averaged over a data base of approximately 150 patients depending on the rhythm class. This is to be contrasted with the previous simpler Gauss-Markov modelling of the ECG, however indicates the adequacy of the use of linear coding approaches.

Acknowledgements

Funding for this work was provided by the Australian Research Council and Telectronics Pacing Systems, Ltd.

References

[Cortes et. al., 1995] C. Cortes, L.D. Jackel, and W. Chiang. Limits on learning machine accuracy imposed by data quality. In *NIPS7*, pages 239–246, 1995.

[Cortes et. al., 1994] C. Cortes, L.D. Jackel, S.A. Solla, V. Vapnik, and J.S. Denker. Learning curves: Asymptotic values and rate of convergence. In *NIPS6*, pages 327–334, 1994.

[Dempster et. al., 1977] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, B(39):1–38, 1977.

[Ghahramani and Jordan, 1994] Z. Ghahramani and M. Jordan. Supervised learning from incomplete data via an em approach. In *NIPS6*, pages 120–127, 1994.

[McLachlan and Basford, 1987] G. McLachlan and K. Basford. *Mixture Models: Inference and applications to clustering.* Marcel Dekker, 1987.

Adaptive Control in Anaesthesia

Alexander Asteroth[1], Knut Möller[1] and Helmut Schwilden[2]

[1] Bonn University – Department of Computer Sience I
Römerstr. 164, D-53117 Bonn, FRG
{aster,moeller}@cs.uni-bonn.de
[2] Clinic for Anaesthesiology
University of Erlangen-Nürnberg, D-91054 Erlangen, FRG

Abstract. Automation of anaesthesia is a complex task but important with respect to patient health, improved quality of narcosis and cost reduction. Furthermore it will enhance our understanding of the complex mechanism underlying anaesthesia.
Classical model based control concepts have been evaluated in the past. Those approaches were limited to univariate process control. As the objective of our studies we want to establish the feasibility of different real-valued reinforcement learning approaches for the task of *multivariate* adaptive control in anaesthesia.
As a first step we present a series of experiments with a naive application of reinforcement learning. The appropriateness is demonstrated in the univariate case. Results are compared to a model based analytical controller.

1 Introduction

Automatic control in anaesthesia is a powerful tool to improve narcosis with respect to:

1. patient specific supply of anaesthetic agents decreases necessary concentration levels to a minimum,
2. support for the anaesthesist in case of temporally delayed effects or nonlinear combination of effects,
3. development of a theoretical foundation of anaesthesia
4. cost reduction

Automatic model based control of volatile anaesthetics exploiting the median EEG-frequency (MEF) was successfully applied in clinical trials [Sch95]. This particular approach was univariate, i.e. the median EEG frequency, MEF was used as input to the controller (state characterization) to adjust the vaporizer setting ($CVap$) as effector[3]. An explicit invertible model

[3] The vaporizer setting influences the concentration of the anaesthetic agent in the inhaled gas, thus controlling the patients anaesthetic depth. Usually EEG median frequency(MEF) is monitored as an indicator for anaesthetic depth. A value of 10Hz MEFcharacterizes a person which is awake while a narcotized person has a $MEF$of less than 3-4Hz.

0-7803-4256-9/97/$10.00

is required capturing the dependence of control parameters and effects. Due to mathematical restrictions this approach can not be generalized to more than one narcotic agent. Also multiple parameters of the patient can not be included.
Reinforcement based learning [Sut88, Tes92, KLM96] has recently matured towards a technique applicable to multidimensional real world tasks. In a project we explore the appropriateness of reinforcement learning systems for automation in anaesthesia.

2 Methods

2.1 Analytic model based closed loop control

Every approach to design a controller requires knowledge about the system to be controlled. Usually this knowledge is represented by an explicit model of the plant. By restricting the number of parameters to one input and one output variable a narcotized patient can approximately be modelled by the $CVap/MEF$ relation.

The applicability of a model of five first order compartments was demonstrated in [Sch95]. It was inverted using the Laplace-transform, thus a controller for the system could be derived.

To fit the patients characteristics the models parameters must be adapted during control. Quality of control relies on these prameter settings. The control scheme is depicted in Fig. 1(a).

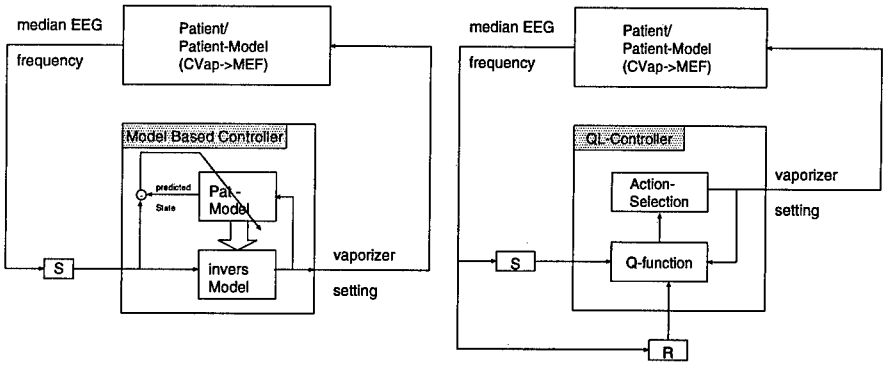


Fig. 1. A schematic overview of the applied controllers. (a) Model based controller with analytically derived invers model. (b) Q learning controller for heuristic adaptive direct control. S denotes sensor evaluation and R reward determination.

Even if this procedure leads to acceptable results in the univariate case it has substantial restrictions. The use of an explicit invertible model hinders the application of more complex and particularly multivariate models.

A control scheme that does not rely on explicit modelling might overcome these limitations.

2.2 Reinforcement learning

Reinforcement learning is a method for heuristic adaptive direct control [Sut88]. Fig.1(b) illustrates Q-learning [Wat89] an instance of reinforcement learning. During learning an evaluation function (Q-function) is approximated that represents the quality of a state action pair with respect to the task. For training the system interacts with it's environment which can either be a real patient or a model of any complexity. Actions (vaporizer settings) to be applied in a certain state are selected probabilistically according to the Boltzmann distribution based on their Q-values.

Thus control does not depend on an invertible model, or even any explicit model at all. It therefore has the potential of controlling multivariate systems which are difficult to model or impossible to invert. In principle reinforcement learning may overcome the limitations of the analytic approach.

A major drawback of reinforecment learning which has to be mentioned here are certain strong requirements in the controlled system. The convergence of Q-learning to an optimal control policy [WD92] is proven if:

1. Q-values are stored by table lookup
2. the controlled system is a Marcov decision process
3. every state-action pair is evaluated arbitrarily often
4. appropriate learning parameters are selected

While 3) and 4) are usually satisfiable 1) and 2) are often violated by real world tasks. If a task is real valued the Q-funtion cannot be stored by lookup tables. Discretization can be used to solve this problem provided that appropriate intervals are chosen. Function approximators such as artificial neural networks can also be used to circumvent the difficulty of choosing a good discretization and to speed up learning by generalization. Both approaches will be used in our experiments.

The problem of non-marcovian processes is eased by using a finite history of state-action information. Unfortunately every increase in dimensionality of the state augments time to convergence by a multiplicative factor if table lookup is used. Additional information for the decision problem has to be traded off against time affordable to explore the exponentially growing state space [TM92]. If the extended state space is of uniform structure function approximators may advantageously be applied.

Despite of the obvious use of function approximation in value function representation, convergence of the learning process can longer be guaranteed. The convergence properties of Q-learning [Ast95] strongly depends on the representation of the Q-function and policy and even if learning converges the final policy might not be optimal [Heg96].

3 Experimental setup

The objective of our learning studies is to develop and to optimize a control policy that stabilizes the MEF at a value of 2.5Hz. For training a number of narcosis sessions are simulated on the basis of the five compartment model, described by [Sch95]. Each of the sessions consistes of 240 minutes of simulated narcosis time.

Uniform noise is added to the $MEF$to keep the problem as realistic as possible. In addition artefacts are simulated to approximate the clinical situation.

As a reference point in our evaluation the analytic controller (i.e. the invers of the model) was used to control the model. In the following figure (Fig.2) an example of an optimal control is shown. Please not that no adaptation of model parameters is necessary and best actions can be determined analytically. In contrast to later experiments regulation is performed every 5 minutes (0.0033 Hz).

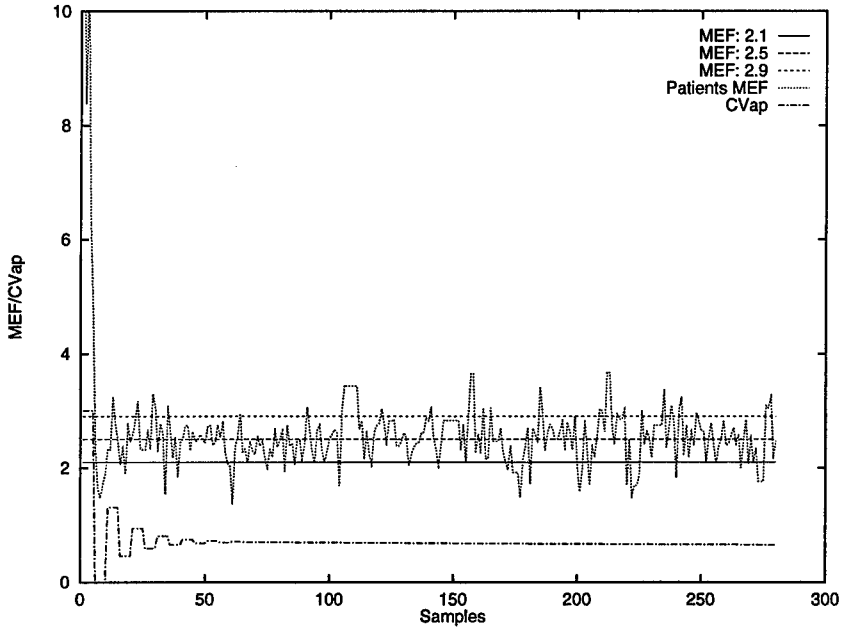


Fig. 2. The control behavior of the model based controller. Control is considered successfull if the MEF stays between 2.1 and 2.9. Parameters were selected in a way that no adjustments in the controller-inherent model were necessary.

The reinforcement controller is trained online, i.e. while controlling the model knowledge is acquired in the Q-approximation and a control policy

develops. Regulation is performed every minute (1/60Hz). As long as a significant improvement is observed training is continued.

Both, the model with and without noise are investigated for training.

Unfortunately the actual MEF alone does not include enough state information to enable proper learning. Therefore a finite history of these values is provided to the system. Because of the exponentially growing state space history size is limited to just a single, the last MEF value. In the next section it will be shown, that this is sufficient to acquire reasonable control policies.

In first experiments table lookup is used to represent the Q-function. The neccesary discretization of the MEF and the $CVap$ settings are realized by 50/20 intervals respectively. Despite this coarse grained discretization reasonable control strategies can be developed. A higher resolution may even lead to better results. In current experiments different function approximators are evaluated.

4 Results

After some training sessions the current policy is evaluated deterministically i.e. by selecting vaporizer settings with maximal estimated utility (Q-value). A sample run is plotted in Fig.3. It shows a similar behavior as the optimal analytic controller although in table (Tab.1) still some differences become apparent.

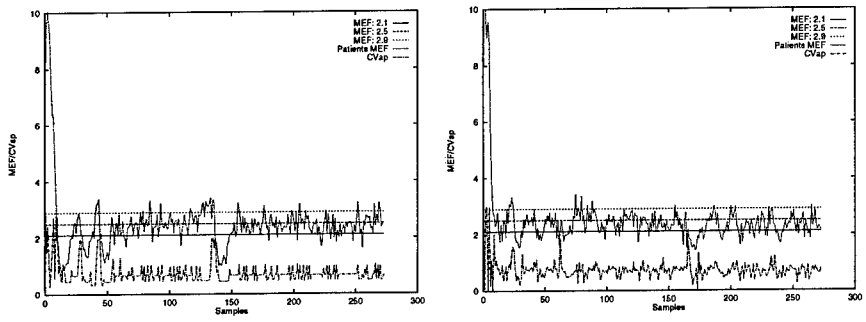


Fig. 3. The control behavior of a reinforcement learning system, with a history of one state, (a) using table look-up (b) using a real valued approximator.

To visualize a learned control policy noise as well as artifact simulation are turned off (cf. Fig.4(a)). Now a deterministic system has to be controlled and the pure control strategy can be examined.

Obviously the learned policy is not able to keep the MEF at a stable level. This is not a principle weakness of the method but is caused by the discretization of the input and output signals. Adjustment of the vaporizer

can not be done precisely because the desired settings usually fall into discretization intervalls. Real valued Q-learning algorithms as shown in Fig.3(b) overcomes this problem.

Policies however do not converge toward a unique stable strategy. Interestingly during learning a variety of strategies can be observed such as an oscillating one depicted in figure 4(b).

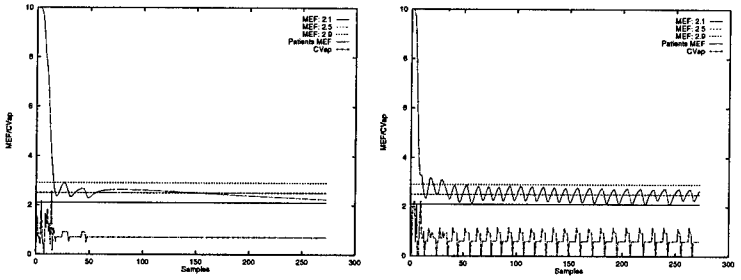


Fig. 4. (a) The same control policy as depicted in figure 3(a) but without noise and artifacts added to the model. (b) The learned oscillating control policy.

To evaluate learned policies four different features are considered: a) the time until a MEF of 2.5 was reached (inital phase of anaesthesia), b) the actual mean value of the median during anaesthesia (steady state), c) the standard deviation of MEF i.e. smoothness of control and d) the average dosis of the anaesthesic applied to the patient. Those values were compared to the corresponding optimal values of the analytic controller and a random controller.

The comparison of mean-value, standard deviation, time to 2.5 and the dosis shows that the reinforcement learning controller based on table look up was not able to reach the peek performance of the analytic controller, but that it's performance is by far better than a reinforcement learning controller without history or even random control methods. A first experiment with a real valued approach shows promising results, that are summarized in the following table.

5 Conclusions and future work

Control of the Q-learning system is –even when using table look-up as representational formalism – within few iterations acceptably close to the analytic solution. In the current experimental setting the performance of the analytic controller represents an upper limit. Optimality though relies on the appropriatness of the invertible model. It has to capture the characteristics of the specific patient to be narcotized. As soon as the model structure does not fit

Method	Time to 2.5	Mean Value	Std. Deviat.	Ø Dosis
Model based	5 min 36 sec	2.49	0.40	0.724
RL real val.	7 min 40 sec	2.51	0.40	0.738
RL w. hist. 1	9 min 40 sec	2.34	0.54	0.742
RL wo. hist.	22 min	1.67	1.07	0.809
Random	-	0.77	1.88	1.224

Table 1. A comparision of some control techniques: i) analytical model based control, ii) reinforcement learning with a real valued representation, iii) reinforcement learning with a one step history using table look-up, iv) reinforcement learning without history, v) a random controller

the real characteristics, e.g. in case of a real patient, learning controllers may perform better, because they are not limited by the model structure.

In this paper it could be demonstrated, that even plain Q-learning is capable of performing near optimal control of univariate anaesthesia. Noise and artifact simulation does not prevent Q-learning from converging. Training is slightly less efficient than with a pure deterministic model but convergence is more robust and resulting strategies are produced more stable.

Still a number of problems are encountered that have to be solved before moving to the more challenging multivariate case.

1. Real valued approaches need to be explored more intensively. Currently a variety of approximators are under investigation including artificial neural network approaches (backprobagation, radial basis function networks, feature maps and extensions to those), memory based techniques and mathematical interpolation methods.
2. The applied state description does not fully capture the models state. The process to be controlled does not possess the desired marcov property. A problem of instablility of the solution arises. First results with the integration of more MEF values and of a past vaporizer setting are promising.
3. Finally a more sophisticated reward function has to be used that e.g. penalizes consumption of anaesthetics. It should as well consider aspects, such as the smoothness of vaporizer settings.

6 Acknowledgements

The authors like to express their gratitude to the "Automatic Anasthesia research group (Prof. Dr. A. Hoeft)" at the University hospital of Bonn University for stimulating discussions.

Support by a grant from DARA (50 WB 9628) to Knut Möller is gratefully acknowledged.

References

[Ast95] Alexander Asteroth. *Über geeignete Repräsentationen der Q-Funktion beim Q-Lernen.* Masters thesis, Dept. of Computer Science, Bonn University, 1995.

[Heg96] Matthias Heger. *The Loss from Imperfect Value Functions in Expectation-Based and Minimax-Based Tasks.* Machine Learning 22, pp. 197-225, 1996.

[KLM96] Leslie P. Kaelbling, Michael L. Littman, Andrew W. Moore. *Reinforcement Learning: A Survey.* Journal of Artificial Intelligence Research 4, pp. 237-285,1996.

[Sch88] Helmut Schwilden. *Qualitative EEG-Analysen von Hypnotika.* Thieme Copythek, Stuttgart, 1988.

[Sch95] H. Schwilden, J. Schüttler. Model-based Adaptive Control of Volatile Anaesthetics by Quantitative EEG. In Control and Automation in Anaesthesia, H. Schwilden, H. Stoeckel (Eds.), Springer Verlag, Berlin, Heidelberg, 1995.

[Sut84] R. S. Sutton. *Temporal credit assignment in reinforcement learning.* Ph.D. thesis, University of Massachusetts, Amherst, MA, 1984.

[Sut88] R. S. Sutton. *Learning to predict by the method of temporal differences.* Machine Learning 3(1), 9-44, 1988.

[Wat89] C. J. C. H Watkins. *Learning from Delayed Rewards.* Ph.D. Thesis, King's College, Cambridge, UK, 1989.

[Tes92] G. Tesauro. *Practical issues on temporal difference learning.* Machine Learning 8, pp. 215-277, 1992.

[WD92] C. J. C. H. Watkins, P. Dayan: Technical Note: *Q-Learning*, Machine Learning, S:279-292, 1992.

[TM92] S. Thrun and K. Möller. *Active exploration in dynamic environments.* In J. Moody, editors, *Advances in Neural Information Processing Systems 4*, San Mateo. Morgan Kaufmann, 1992.

WAVELET CHARACTERISTICS OF EARLY VISION

Geoffrey Brooks
WL/MNGA, Eglin AFB FL 32542

ABSTRACT

Recent advances in wavelet theory are affording great opportunities for signal processing applications. Natural neuronal networks exhibit wavelet behavior from which structural and functional paradigms could be exploited for machine-vision applications. Provided here is a summary of the ways vertebrate vision systems naturally exhibit wavelet characteristics.

INTRODUCTION

Wavelets can be generally defined as little waves that start and stop and originate from a single basic function [18]. The Wavelet Transform decomposes signals into their wavelet components as the Fourier Transform decomposes signals into their frequency components. Fourier time-frequency representations are presented here as an introduction to the more general wavelet representations.

Early vision can be defined as the processes that recover the properties of object surfaces from 2D intensity arrays [6]. The structure and function of natural vision systems exhibit wavelet characteristics in many ways. The focus here is on vertebrate vision information pathways that begin in the retina and terminate in cortical processing stages. Many of these concepts are also common in insect vision.

CONVENTIONAL TIME-FREQUENCY REPRESENTATIONS

Fourier developed a series of weighted sine and cosine terms to represent a periodic waveform $f(t)$, with period $T = 2\pi/\omega_0$, where ω_0 is the fundamental radian frequency. The Fourier Series is an infinite sum of components, weighted by coefficients a_n, at integer, n, multiples of ω_0 [15,20]:

$$f(t) = \sum a_n e^{-jn\omega_0 t} \qquad (1)$$

As T increases, ω_0 decreases along with the frequency spacing between Fourier Series components (or terms). As $T \rightarrow \infty$, the frequency distance between components becomes infinitesimal so that $\omega_0 \rightarrow \omega$, a continuous variable. The

solution for coefficients, when integrated over one sample period, becomes the Fourier Transform:

$$(Ff)(\omega,t) = \int dt f(t)e^{-j\omega t} = \langle f(t), e^{j\omega t} \rangle , \qquad (2)$$

where $\langle \rangle$ denotes the inner product.

For real applications, functions of time are analyzed in finite intervals. To transform $f(t)$ into its frequency components, the signal is assumed periodic in the sampling interval. The duration of this interval becomes the fundamental period T, which also defines the frequency resolution $\Delta\omega = \omega_0$. For long sampling intervals (large T), frequency details are well resolved (small $\Delta\omega$), but the location in time of specific events within T is not known. Compensating for this by reducing T around a specific time t_0 results in large $\Delta\omega$, hence an uncertainty of specific frequency components. This tradeoff between time and frequency resolution is known as the *time-frequency uncertainty.*

An alternative to a specified duration of $f(t)$ is to window $f(t)$ with a Gaussian-like function, $g(t)$, centered at time t_0. This windowing operator, T^{win}, serves to weight the signal closest to time t_0 more heavily than the signal farther away in time while maintaining a reasonably large T for adequate frequency resolution. In addition, the decay of T^{win} reduces the undesirable effects of *Gibb's phenomenon* at the edges [15]. The result is a Fourier transform that is a function of t_0, called the *Windowed Fourier Transform*:

$$(T^{win}f)(\omega,t_0) = \int dt f(t)g(t-t_0)e^{-j\omega t} = \langle f(t), g(t-t_0)e^{j\omega t} \rangle \qquad (3)$$

The *Discrete* Windowed Fourier Transform results in integer m sampled spectral components ($m\omega_0$) derived from integer n sampled time intervals (nt) of a signal. It is given as [7]

$$T^{win}_{m,n}f = \int dt f(t)g(t-nt_0)e^{-jm\omega_0 t} = \langle f(t), G_{m,n}(t) \rangle , \qquad (4)$$

where $G_{m,n}(t) = g(t-nt_0)e^{-jm\omega_0 t}$.

WAVELET TIME-FREQUENCY REPRESENTATIONS

Given an appropriate function space, the set of functions generated by recursive applications of the *scaling function* to the *mother wavelet* constitutes a set of basis functions called a *wavelet series* [4]. The *Continuous Wavelet Transform* (CWT) is similar to the Fourier Transform (or Windowed Fourier Transform) in that both involve an inner product of an input function with a scaling function. The CWT is given as [7]

$$(T^{wav}f)(a,b) = |a|^{-1/2}\int dt f(t)\psi^*((t-b)/a)$$
$$= |a|^{-1/2} <f(t),\psi((t-b)/a)>, \quad (5)$$

Variable a, the *dilation parameter*, controls the interval of compact support, or duration, for the wavelet $\psi^{a,b} = \psi((t-b)/a)$. Variable b, the *translation parameter*, determines where in time the wavelet exists. The *Discrete Wavelet Transform* (DWT) is a collection of samples of the CWT, sampled at m incremental dilations and n incremental translations, given as [7]

$$T^{wav}{}_{m,n}(f) = |a_0|^{-m/2}\int dt f(t)\psi^*(a_0^{-m}t - nb_0)$$
$$= |a_0|^{-m/2} <f(t),\psi(a_0^{-m}t - nb_0)> \quad (6)$$

For a given application, the optimal basis functions, or wavelet series components, must be chosen as well as the optimal scaling and translation parameters. These selections make the application of wavelet transforms more complicated than that of Fourier transforms.

Consider a DWT implemented with unity dilation, $a_0 = 1$, and a mother wavelet that is a windowed complex exponential, $\psi^{a_0,b_0} = g(t-nt_0)e^{-jm\omega_0 t}$; in this case the DWT becomes the DFT. Using unity dilation defeats the utility of allowing for a better balance between frequency resolution at lower frequency components and time resolution at higher frequency components. However, this demonstrates that the Fourier Transform can be thought of as a special case of the Wavelet Transform.

STRUCTURE AND FUNCTION OF EARLY VISION

Natural vision filtering begins with photonic refraction through the cornea and lens. The incoming light then passes through the vitreous humor and retinal cell tissue, and is focused onto a *photoreceptor mosaic* surface. Photonic energy is converted to electronic charge in the photopigment discs of the photoreceptors (rods and cones). The photoreceptors, with the help of a layer of *horizontal* cells, spread the charge in space and time within a local neighborhood of other receptors. The spread charge and photoreceptor charge are both available at the root of the photoreceptor, at the *triad synapse*. The *bipolar* cells connect to triad synapses and presumably activate signals proportional to the difference between the photoreceptor input and the horizontal cell input. Both polarities exist, called *on*-bipolars or *off*-bipolars, which respond to light and darkness respectively [11,14,23].

The photoreceptor charge is influenced by gap junctions between adjacent photoreceptors. The response from a photoreceptor aggregate can be

modeled as a spatial-temporal Gaussian with a small variance. The input from the neighboring aggregate of horizontal cells can be modeled with a similar Gaussian with a larger variance. The differencing function results in the *difference-of-Gaussian* (DOG) filter operation, resulting in a center-surround antagonistic receptive field profile. DOG and *Laplacian-of-Gaussian* (LOG) functions have been used to model the bipolar cell output [14].

The analog charge information in the retina is funneled into information pathways as it is channeled from the mosaic plane to the optic nerve [3]. The channels that exist there are the *rod* channel, initiated by rod bipolars, the *parvocellular pathway* (PP) and the *magnocellular pathway* (MP), the latter two initiated by cone bipolars [10]. Both the PP and the MP exhibit center-surround antagonistic receptive fields. PP cones are tightly connected, responding to small receptive fields, while the MP cones are more loosely connected (together with rod inputs), responding to large receptive fields.

The MP and PP perform separate spatial band-pass filtering, provide color and intensity information, and also provide temporal response channels, as illustrated in Figure 1. A relatively high degree of acuity is achieved in each domain from these few filters. The MP is sensitive to low spatial frequencies and broad color intensities, which provide basic information of the objects in the image. The PP is known to be sensitive to higher spatial frequencies and chromatic differences, which add detail and resolution [17]. In the color domain, the PP provides color opponency and thus spectral specificity, and the MP provides color non-opponency and thus overall intensity [9,12]. In the time domain, the PP provides slowly varying dynamics, while the MP provides transient responses to image dynamics.

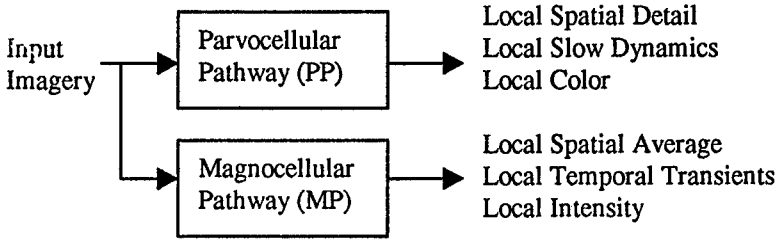


Figure 1. Natural Vision Information Channels.
The color opponent PP responds to spatial detail, slowly-varying image dynamics, and chromatic detail. The color non-opponent MP responds to spatial averages, rapid transients, and intensity variations.

WAVELET FEATURES INHERENT IN VISION PROCESSING

Wavelet bases can be subdivided into *orthogonal* or *nonorthogonal* and *complete* or *noncomplete* categories [21]. A set of basis functions is *orthogonal* if the inner product of any two different basis functions is zero, and *complete* if no non-zero function in the space is orthogonal to every basis vector [4,22]. Orthogonality and completeness ensure a unique transformed representation of each function within the given space. These are the general requirements for the selection of wavelet bases in compression applications.

However, biological systems are not concerned with information storage for perfect reconstruction. Any machine-vision application requiring some action to be taken based on an understanding of the image content will also fit this general description. In fact, many biological processing functions are considered to be non-orthogonal [8]. The task is processing information in order to take some action, not processing information for later reconstruction. The redundancy of vision filters is balanced by the need for efficiency, simplicity, and robustness. Information redundancy results in unnecessary hardware and interconnections, but often redundancy may be required to sufficiently span the information space inherent in the environment. The cost of supporting the redundancy may be less significant than the benefit of using simpler processing elements that degrade gracefully.

Wavelet Filter Banks and Vision Pathways

The MP and PP decompose the natural input image into local average and local detail components, respectively. Images can also be decomposed into wavelet components using *quadrature mirror filtering* (QMF), resulting in a series of averaging components and another series of detailing components [7,19]. QMF is a special case of *subband coding*, where filtered components represent the lower and upper frequency halves of the original signal bandwidth. If the analyzing filter coefficients are symmetric, then the synthesizing components are mirrored with respect to the half-band value, thus the term *quadrature mirror*. A variety of applications have emerged from the remarkable QMF reconstruction capabilities [13,16,18].

Vision pathways (MP and PP) and QMF filter banks both therefore break up the input image signal into high and low frequency components. Recent spectral analysis of Gaussian-derived vision models indicate a remarkable retention of information in spite of the nonorthogonal nature of vision filters [2,3].

Examples of Waveform Translation and Dilation in Vision Systems

As described previously, a wavelet basis is generated by recursively applying a scaling function to a mother wavelet [4,7]. The essential characteristics of the

wavelet scaling function are the translation and dilation parameters of (5) and (6). Structural and functional subunits of vision systems also tend to replicate a mother function with scaling and translational variations. These variations are functions of other parameters such as time, gaze direction, and *eccentricity*, the distance from the center of the retina.

Photonic spreading is a function of the finite aperture, optical imperfections, and photonic wavelength [10]. A Gaussian mother wavelet of a carefully chosen variance can be chosen to model the point spread function at the retina center (zero eccentricity) at a selected visible frequency. The complete point spread function can be modeled as a two-dimensional scaling of this mother function with respect to both eccentricity and photonic frequency.

The photoreceptor mosaic includes rods and three cone types, called *L*, *M*, and *S* for "long", "medium", and "short" visible wavelength peaks in their spectral absorption curves. Sampling in the center of the retina is limited to L and M cells, providing a significantly higher spatial resolution as compared to the periphery [10,11]. For a given viewing position, a mental model of an environment is built from a set of high resolution samples from different directions. These translations of the mosaic function are the result of eye movements [5].

Due to scaling with frequency, higher frequency wavelet components typically have smaller regions of support. The sampling density of M and L cone cells is greatly reduced as a function of increasing eccentricity. This is due to increasing cell sizes and also by the introduction of rod cells and S cells in the outer regions. A cellular receptive field function representing the envelope of available photonic energy stimulating either M or L cones will thus include an increasing scale factor with eccentricity.

The initial electronic processing stages of early vision are characterized by DOG filtering operations of the MP and PP. The spatial extent of both the MP and PP receptive fields increase significantly with eccentricity [10]. A DOG mother function can be scaled with respect to eccentricity to properly model vision pathways at different parts of the retina.

Efficient Use of Basis Functions

A common theme among space, time, and color domains is the minimal use of basis functions. There are essentially four chromatic detector types, three temporal channels, and three spatial channels. Therefore, natural neuronal systems tend to use efficient combinations of only a few filters to accomplish a high degree of acuity in each domain. QMF signal reconstruction capability is a practical demonstration of extracting spectral detail from only two filters. The behavior of such synthetic applications may lead to a deeper understanding of natural phenomena.

ACKNOWLEDGMENTS

This effort was performed in-house at the Advanced Guidance Division of the Wright Laboratory Armament Directorate located at Eglin AFB FL. This research leverages recent doctoral research at the Department of Electrical Engineering of the FAMU-FSU College of Engineering in Tallahassee FL. Comments and collaboration are encouraged[a].

REFERENCES

[1] Brooks, Geoffrey, "Comparison of vision and conventional wavelet filters," Proceedings of the SPIE, vol. 3078, no. 48, 1997.
[2] Brooks, Geoffrey, "Biologically-inspired analog wavelet analyzers," Proc. of the SPIE, vol. 3016, no.8, 1997.
[3] Brooks, Geoffrey, Modeling Retinal Processing using Wavelet Theory, Ph.D. Dissertation, Florida State University, 1996.
[4] Chui, Charles, An Introduction to Wavelets, Academic Press, 1994.
[5] Churchland, P., Ramachandran, V., and Sejnowski, T., "A critique of pure vision," in Koch, C., and Davis, J., editors, Large-Scale Neuronal Theories of the Brain, MIT Press, Cambridge, Mass., 1994.
[6] Dagnelie, Gislin, and Massof, Robert, "Toward an Artificial Eye," IEEE Spectrum Magazine, May, 1996.
[7] Daubechies, Ingrid, Ten Lectures on Wavelets, Society for Industrial and Applied Mathematics, Philadelphia, 1992.
[8] Daugman, John, "Complete Discrete 2-D Gabor Transforms by Neural Networks for Image Analysis and Compression," IEEE Transactions on Acoustic, Speech, and Signal Processing, vol. 36, no. 7, July 1988.
[9] De Valois, R. L., & De Valois, K. K., "A multi-stage color model," Vision Research, vol. 33, no. 8, 1993.
[10] De Valois, R. L., & De Valois, K. K., Spatial Vision, Oxford University Press, New York, NY, 1988.
[11] Dowling, John E., The Retina: An Approachable Part of the Brain, Harvard University Press, Cambridge, Mass., 1987.
[12] Gath, S. Lee, "Model for color vision and light adaptation," Journal of the Optical Society of America, vol. 8, no. 6, 1991.
[13] Mallat, Stephane, "A Theory for Multiresolution Signal Decomposition: The Wavelet Rep.," IEEE Transactions on PAMI, vol. 11, no. 7, 1989.
[14] Marr, David, Vision, W. H. Freeman and Company, New York, 1982.
[15] Oppenheim, Alan, and Schafer, Ronald, Discrete-time Signal Processing, Prentice Hall, Englewood Cliffs, NJ, 1989.
[16] Rioul, Oliver, and Vetterli, Martin, "Wavelets and Signal Processing," IEEE Signal Processing Magazine, October, 1991.

[17] Sherman, S. M., "Parallel W-, X-, and Y-cell pathways in the cat: a model for visual function," in Rose, D., and Dobson, V., eds, Models of the Visual Cortex, J. Wiley & Sons, New York, NY, 1985.
[18] Strang, Gilbert, "Wavelets," American Scientist, vol. 82, May-June, 1994.
[19] Strang, Gilbert, and Nguyen, Truong, Wavelets and Filter Banks, Wellesley-Cambridge Press, Wellesley, MA, 1996.
[20] Strum, Robert, and Kirk, Donald, First Principles of Discrete Systems and Digital Signal Processing, Addison-Wesley, New York, 1989.
[21] Szu, Harold, "Wavelet theory and applications," short course notes for the SPIE AeroSense95 Conference, Orlando, FL, April 1995.
[22] Walter, Gilbert, Wavelets and Other Orthogonal Systems with Applications, CRC Press, Inc., Boca Raton, FL, 1994.
[23] Werblin, Frank S. and Teeters, Jeffrey L., "Real time simulation of the retina allowing visualization of each processing stage," Proceedings of the SPIE vol. 1472, 1991.

[a] Further author information: WL/MNGA, Eglin ABF, FL 32542 Email: brooksg@eglin.af.mil

ALGORITHMS

Invited lecture

Neural Networks and Gradient-Based Learning in OCR

Yann LeCun

AT&T Labs - Research
Red Bank, NJ

A large proportion of today's commercial Optical Character Recognition systems (OCR) and Handwriting Recognition Systems (HWR) use neural networks at the core of the recognition engine. Comparisons on standard databases show that Neural Networks, particularly multi-layer networks, offer a good combination of speed, generality, simplicity, and flexibility. They are also particularly well-suited for the large input dimension required for shape recognition tasks such as character recognition.

Neural Networks and Machine Learning have become indispensable ingredient in the design of OCR/HWR systems. Those systems are generally built as a cascade of independent modules including: line and word locators, character segmenters, feature extractors, character recognizers, and language models. However, in most cases, only the character recognizer is trainable. We describe a new learning paradigm called Graph Transformer Networks that allows all the modules in such a system to be trained simultaneously so as to maximize a global performance measure. Each module, called a Graph Transformer, takes graphs as input and produces graphs as output. The arcs on the graphs carry numerical information (scalars or vectors) such as images, scores, and class labels. A gradient-based learning procedure can be used to train the parameters of the modules so as to maximize a global objective function.

Graph Transformer modules offer much increased flexibility over traditional gradient-based learning systems such as multilayer neural networks that communicate their states and gradients via fixed-size vectors. A complete system based on this concept for reading handwritten and printed bank checks is described. It contains hundreds of thousands of trainable parameters and combines convolutional neural network character recognizers with graph-based stochastic models trained cooperatively at the document level. It is deployed commercially and reads million of business and personal checks per month with record accuracy.

The Gamma MLP – Using Multiple Temporal Resolutions for Improved Classification

Steve Lawrence[1], Andrew D. Back[2], Ah Chung Tsoi[3], C. Lee Giles[1]

[1] NEC Research Institute, 4 Independence Way, Princeton, NJ 08540
[2] The Institute of Physical and Chemical Research (RIKEN), Japan
[3] Faculty of Informatics, University of Wollongong, NSW 2522 Australia

Abstract

We have previously introduced the Gamma MLP which is defined as an MLP with the usual synaptic weights replaced by gamma filters and associated gain terms throughout all layers. In this paper we apply the Gamma MLP to a larger scale speech phoneme recognition problem, analyze the operation of the network, and investigate why the Gamma MLP can perform better than alternatives. The Gamma MLP is capable of employing multiple temporal resolutions (the temporal resolution is defined here, as per de Vries and Principe, as the number of parameters of freedom (i.e. the number of tap variables) per unit of time in the gamma memory – this is equal to the gamma memory μ parameter as detailed in the paper). Multiple temporal resolutions may be advantageous for certain problems, e.g. different resolutions may be optimal for extracting different features from the input data. For the problem in this paper, the Gamma MLP is observed to use a large range of temporal resolutions. In comparison, TDNN networks typically use only a single temporal resolution. Further motivation for the Gamma MLP is related to the "curse of dimensionality" and the ability of the Gamma MLP to trade off temporal resolution for memory depth, and therefore increase memory depth without increasing the dimensionality of the network. The IIR MLP is a more general version of the Gamma MLP – however the IIR MLP performs poorly for the problem in this paper. Investigation suggests that the error surface of the Gamma MLP is more suitable for gradient descent training than the error surface of the IIR MLP.

0-7803-4256-9/97/$10.00

1 Introduction

Machine learning models used for speech recognition are required to account for a high degree of variability in the data (e.g. acoustic variability, *within-speaker* variability, *across-speaker* variability, and phonetic variability). For phoneme recognition, methods of addressing these variabilities include using larger datasets and using models which take into account greater context of the acoustic signal. However, taking into account greater context typically leads to larger models. The amount of training data required for accurate estimation of class distributions can increase significantly when the input dimensionality increases (cf. the "curse of dimensionality" [6])[1]. As the complexity of the desired target function for a given problem increases while the amount of data remains constant, it becomes increasingly problematic to estimate the target function from finite data due to the ill-posed nature of the problem – many of the models which fit the training data closely do not generalize well to unseen data. In order to reduce the difficulty with trying to approximate a function which is too complex for the available data, we often consider looking for a hierarchical solution where initial layers extract features which identify higher level attributes of the data which enhance generalization. These features can be extracted manually, or automatically. The Gamma MLP considers a transformation for the inputs to each node and aims to optimize the transformation for each node individually in order to improve performance. The process can be thought of as automatic feature extraction (if the optimal transformations were known beforehand then those transformations could be used to extract new features from the data).

2 The Gamma Filter

Infinite Impulse Response (IIR) filters have a significant advantage over Finite Impulse Response (FIR) filters in signal processing: the length of the impulse response is uncoupled from the number of filter parameters. The length of the impulse response is related to the memory depth[2] of a system, and hence IIR filters allow a greater memory depth than FIR filters of the same order. However, IIR filters are not widely used in adaptive signal processing [9]. This may be attributed to the fact that a) there may be instability during training and b) the gradient descent training procedures are not guaranteed to locate the global optimum in the possibly non-convex error surface [11].

[1] Additionally, increases in the "complexity" of the desired target function may make gradient descent optimization more difficult – training algorithms may take longer to converge or become "stuck" in local minima or "plateaus" which are increasingly poor compared to the global optimum.

[2] A greater memory depth implies that the model can retain past information for a longer time.

The use of *gamma* filters as a memory structure at the input of an otherwise standard MLP network was proposed by de Vries and Principe [5]. The gamma filter, a special case of an IIR filter, is designed to retain the uncoupling of memory depth to the number of parameters provided by IIR filters, but to have simple stability conditions. The output of a neuron in a multilayer perceptron is computed using[3] $y_k^l = f\left(\sum_{i=0}^{N_{l-1}} w_{ki}^l y_i^{l-1}\right)$. The addition of short term memory with delays was considered by de Vries and Principe [5]: $y_k^l = f\left(\sum_{i=0}^{N_{l-1}} \sum_{j=0}^{K} g_{kij}^l(t-j) y_i^{l-1}(t-j)\right)$ where $g_{kij}^l(t) = \frac{\mu_{ki}^l}{(j-1)!} t^{j-1} e^{-\mu_{ki}^l t}$, $j = 1, 2, \ldots, K$. The depth of the memory is controlled by μ, and K is the order of the filter. For the discrete time case, de Vries and Principe [5] obtain the following recurrence relation:

$$z_j(t) = \begin{cases} x(t), & j = 0 \\ (1-\mu) z_j(t-1) + \mu z_{j-1}(t-1), & j = 1, 2, \ldots, K \end{cases} \quad (1)$$

where $x(t)$ is the filter input and $z_j(t)$ are the filter outputs. For $\mu < 1$ the gamma filter may be considered as a low pass filter. For $\mu = 1$, the memory is a tapped delay line corresponding to the memory structure in an FIR MLP (An MLP where the weights are replaced by FIR filters and optional gain terms [2]) or a TDNN. For $\mu < 1$ the gamma memory structure implements a tapped dispersive delay line where the degree of dispersion is controlled by μ.

de Vries and Principe [9] define the temporal resolution, R, of a gamma memory structure as the number of parameters of freedom (i.e. the number of tap variables) per unit of time in the filter memory: $R = K/D = \mu$ where D is the memory depth of the structure (the temporal mean value of the impulse response of the last tap) [10]: $D = K/\mu$. When $\mu = 1$, the memory depth is equal to the order of the memory, K. The memory depth increases when $\mu < 1$, and the temporal resolution decreases, i.e. the gamma memory can trade resolution for memory depth. Therefore the gamma memory can be used to create models which can take into account greater context with fewer parameters (without resorting to the use of a single low temporal resolution) in comparison to TDNN or FIR MLP models.

[3] where y_k^l is the output of neuron k in layer l, N_l is the number of neurons in layer l, w_{ki}^l is the weight connecting neuron k in layer l to neuron i in layer $l-1$, $y_0^l = 1$ (bias), and f is commonly a sigmoid function.

3 The Gamma MLP

3.1 Motivation

The *focused gamma network* which uses the gamma memory as a preprocessing layer for a standard MLP has been proposed by de Vries and Principe [5]. This network allows for the use of only one temporal resolution per input. However, it may be desirable to use multiple temporal resolutions (e.g. different resolutions may be optimal for extracting different features or for classifying different phonemes). The Gamma MLP is similar to a standard MLP except every synapse contains a gamma memory structure and a gain factor. The temporal resolution of the memory in each synapse is adjusted separately. Therefore, in contrast with the focused gamma network, the Gamma MLP is able to use multiple temporal resolutions. Additionally, the Gamma MLP can contain gamma memory structures in all layers of the network.

Other motivation for the Gamma MLP can be seen with comparison to TDNN, FIR MLP and IIR MLP (An MLP where the weights are replaced by IIR filters and optional gain terms [1]) models. In comparison to the TDNN and FIR MLP models, the Gamma MLP may provide improved performance because it allows temporal resolution to be traded for memory depth, i.e. for a system of given dimensionality, the Gamma MLP can employ filters with a greater memory depth. Additionally, in comparison with the IIR MLP, the Gamma MLP may be significantly easier to train, which is discussed further in section 5.

3.2 Definition

Definition 1 A Gamma MLP with L layers excluding the input layer $(0, 1, ..., L)$, gamma filters of order K, and $N_0, N_1, ..., N_L$ neurons per layer, is defined as:

$$y_k^l(t) = f\left(x_k^l(t)\right) \tag{2}$$

$$x_k^l(t) = \sum_{i=0}^{N_{l-1}} c_{ki}^l(t) \sum_{j=0}^{K} w_{kij}^l(t) z_{kij}^l(t) \tag{3}$$

$$z_{kij}^l(t) = \begin{cases} (1-\mu_{ki}^l(t)) z_{kij}^l(t-1) + \mu_{ki}^l(t) z_{ki(j-1)}^l(t-1), & 1 \le j \le K \\ y_i^{l-1}(t), & j = 0 \end{cases} \tag{4}$$

where $y_k^l(t)$ is the output of neuron k in layer l at time t, c_{ki}^l = synaptic gain, $\mathrm{f}(\alpha) = \tanh(\alpha) = (e^{\alpha/2} - e^{-\alpha/2})/(e^{\alpha/2} + e^{-\alpha/2})$, $k = 1, 2, ..., N_l$ (neuron index), $l = 0, 1, ..., L$ (layer), and $z_{kij}^l|_{i=0} = 1, w_{kij}^l|_{i=0, j\neq 0} = 0, c_{kij}^l|_{i=0} = 1$ (bias).

□

A Gamma MLP is defined as a multilayer perceptron where every synapse contains a gamma filter and a gain term (introduced in [7]), as shown in the definition above. The Gamma MLP is therefore a special case of the IIR MLP [1]. The motivation behind the inclusion of the gain term is discussed in section 5. A separate μ parameter is used for each filter. Gradient descent update equations for the Gamma MLP are given in [7]. In practice, it is often desirable to restrict the Gamma MLP structure by using Gamma filter only in the first layer and/or not using the synaptic gain terms (c_{ki}^l), as is also the case for FIR and IIR MLP networks.

4 Phoneme Recognition

4.1 Task Details

Our data consists of the "sa" sentences spoken by male members of demographic region 3 in the TIMIT database. There are 79 speakers. The problem is therefore speaker independent phoneme prediction. The speakers in the training and test sets do not overlap.

The raw speech data was preprocessed into a sequence of frames using PLP. The analysis window (frame) was 20 ms. Each succeeding frame overlapped with the preceding frame by 10 ms. 9 PLP coefficients plus the signal power were extracted and used as features describing each frame of data. The difference between the current and previous frames was added to the input vectors, as is commonly done [4]. Periods of silence before and after the sentences were reduced to two frames in order to limit any skew of the results caused by a disproportionate percentage of silence frames.

The models had 40 outputs corresponding to the 40 phonemes[4]. The FIR and gamma filter orders were 4 (5 taps), and the TDNN model had an input window of 5 steps in time. The training set contained 10,000 frames, the test set and validation sets contained 5,000 frames, and the networks had 40 hidden nodes. The networks were trained for 200,000 updates. We used standard backpropagation with stochastic update. The tanh activation function was used. A "search then converge" learning rate schedule was used with an initial learning rate of 0.1 for the μ parameters and 0.2 for all other parameters.

[4]The TIMIT allophones were converted to the standard 40 phoneme set [8].

4.2 Results

Results are presented for frame level phoneme recognition, i.e. for each frame the recognizer predicts the current phoneme. A few observations regarding the expected results: guessing would result in 97.5% error, coarticulation[5] makes the task difficult, and the possibility of zero error would not be expected due to inevitable difficulty and errors in the phoneme labelling.

The frequencies of the forty classes varies significantly, and it was found that all models had a tendency to "ignore" the rarer phonemes [3] due to biases inherent in the neural network architecture and training algorithm. We therefore employed a scaling technique whereby weight updates are scaled on a class by class basis. The amount of scaling is varied using a control parameter, c_s, from none ($c_s = 0$) to scaling according to the prior probabilities of the classes ($c_s = 1$). Yaeger et al. have recently introduced a very similar technique which they call "frequency balancing" [12].

Reporting results in terms of the percentage of correct classifications can be misleading when the frequency of the individual classes varies significantly (e.g. a relatively low error rate may be achieved by a network which ignores low frequency classes). For this reason, results are reported here in terms of the MSSE which is defined as: MSSE $= \frac{1}{N_c} \sum_{i=1}^{N_c} (1 - S_i)^2$ where N_c = the number of classes and S_i = the sensitivity of class i. The sensitivity of a class is defined as the proportion of events labelled as that class which are correctly detected. This criterion was chosen because each class is given equal importance and the square causes lower individual sensitivities to be penalized more (e.g. for a two class problem, class sensitivities of 100% and 0% produce a higher MSSE than sensitivities of 50% and 50%).

Figure 1 shows the results for the Gamma MLP, FIR MLP, and TDNN networks. The degree of scaling, c_s, was varied from 0 to 1. Five trials were performed in each case. The FIR MLP and Gamma MLP networks contained filters in both layers. The Gamma MLP contained synaptic gains, however the FIR MLP was found to perform significantly better without the synaptic gains for this problem. Scaling with $c_s = 0.75$ resulted in the best performance for each of the networks and, therefore, scaling with $c_s = 0.75$ was used for the later results.

Results for the IIR MLP are not shown because it was not possible to obtain significant convergence. Theoretically, the IIR MLP model is the most powerful model used here (in the sense that it can represent a greater variety of computational structures than the other networks with the same number of hidden nodes). In particular, the Gamma MLP is a special case of the IIR MLP. Although the IIR MLP is prone

[5]Coarticulation refers to changes in the way a speech segment is articulated depending on previous (backward coarticulation) and following segments (forward coarticulation).

to stability problems, the stability of the model can and was controlled in the simulations performed here (by reflecting poles that move outside the unit circle back inside). The most obvious hypothesis for the difficulty in training the model is related to the error surface and the nature of gradient descent. It is expected that the error surface of the IIR MLP presents greater difficulty to gradient descent optimization. This is discussed further in the next section.

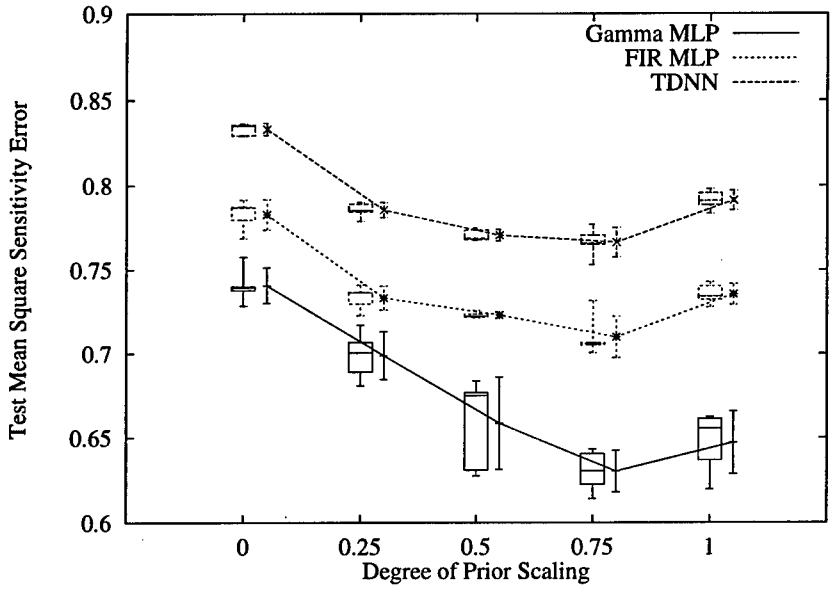


Figure 1. Test MSSE results as the degree of scaling is modified. The best error corresponds to a scaling degree of 0.75 for each network type. At each point, box-whiskers plots are shown on the left and the mean plus and minus one standard deviation is shown on the right. Five trials were performed in each case.

5 Discussion

The Gamma MLP may perform better than the standard TDNN and the FIR MLP for speech recognition because the gamma filtering operation allows processing the input data using multiple temporal resolutions. The Gamma MLP can therefore account for more past history of the signal for a system of a given order (without resorting to the use of a single low temporal resolution). Figure 2 shows the distribution of the gamma μ parameters in a typical trained Gamma MLP. It can be seen that

a range of μ parameters, and therefore a range of temporal resolutions, is employed by the network.

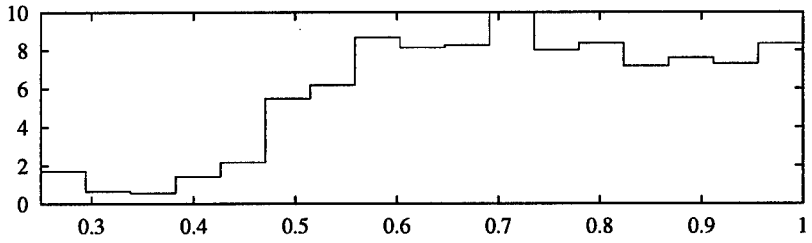


Figure 2. The final distribution of the gamma μ parameters for a sample Gamma MLP.

The Gamma MLP often performs better when using the synaptic gain terms. This improvement may be considered non-intuitive to many – the synaptic gains add degrees of freedom, but no additional representational power. However, the error surface will be different in each case, and results indicate that the surface for the synaptic gains case can often be more amenable to gradient descent.

For the problem considered here, the Gamma MLP performs significantly better than the IIR MLP although the Gamma MLP is a special case of the IIR MLP. It is reasonable to believe that the IIR MLP could perform as well as, or possibly better than, the Gamma MLP, but in practice it is difficult to make it do so for the problem considered here. Figure 3 shows sample plots of the error surface for Gamma and IIR MLP networks. In order to reduce computational expense and use networks with fewer parameters to aid visualization, a simpler task has been chosen. The task is Mackey-Glass prediction using networks that contain only five hidden nodes (the order of the filters was 4, the initial learning rate was 0.1, the training, test, and validation sets contained 500 points, and 100,000 stochastic updates were performed in each case). Even with such small networks, the error surface has many dimensions making visualization difficult. Each plot in the figures is with respect to two randomly chosen dimensions. In each case, the center of the plot corresponds to the values of the parameters after training and the range of each parameter on the plot is 8. The NMSE was evaluated at 225 points equally spaced in a grid. For the IIR MLP, a greater percentage of "flat spots" and complex surfaces can be observed. On average, the error surface for the IIR MLP appears to be less suitable for gradient descent optimization, reinforcing the conclusion that the poorer performance of the IIR MLP is due to optimization being more difficult. Hence, in using the Gamma MLP instead of the IIR MLP, we are trading off computational capacity for easier training. The test NMSE results for 20 simulations each using these networks show that the best performing IIR MLP was only slightly worse than

the best performing Gamma MLP. However, the Gamma MLP was significantly better on average (NMSE of 0.0341 versus 0.185 for the IIR MLP).

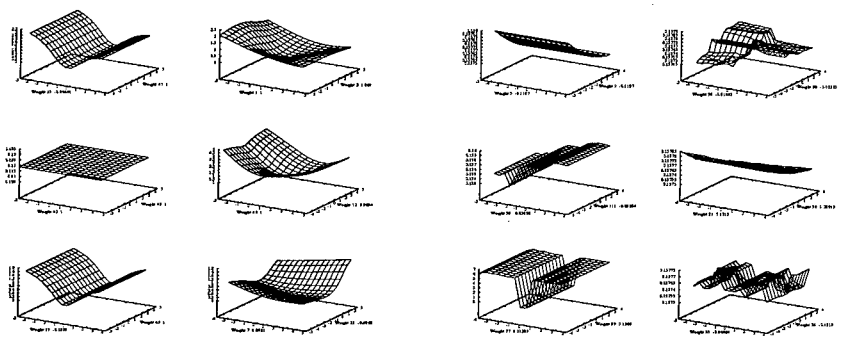


Figure 3. Error surface plots for a sample Gamma MLP (left two columns) and a sample IIR MLP (right two columns). Each plot is with respect to two randomly chosen dimensions. In each case, the center of the plot corresponds to the values of the parameters after training. The z-axis scale varies from plot to plot in order to show the qualitative aspects of the surface (the plots only cover variation in two dimensions and are only plotted around one point in weight space, therefore quantitative conclusions should be drawn from the final NMSE results). From many of these plots we have observed that there is a greater percentage of "flat spots" and complex surfaces for the IIR MLP.

6 Conclusions

We have applied the Gamma MLP to a speech phoneme recognition problem, analyzed the operation of the network, and investigated why the Gamma MLP can perform better than alternatives. The Gamma MLP is capable of employing multiple temporal resolutions, which may be advantageous for certain problems, e.g. different resolutions may be optimal for extracting different features from the input data. For the problem in this paper, the Gamma MLP is observed to use a large range of temporal resolutions. In comparison, TDNN networks typically use only a single temporal resolution. The Gamma MLP is able to trade off temporal resolution for memory depth, and therefore increase memory depth without increasing the dimensionality of the network (or using a single low temporal resolution). The IIR MLP is a more general version of the Gamma MLP – however the IIR MLP performed poorly for the problem in this paper. Investigation suggested that the error surface of

the Gamma MLP is more suitable for gradient descent training than the error surface of the IIR MLP.

References

[1] A.D. Back. *New Techniques for Nonlinear System Identification: A Rapprochement Between Neural Networks and Linear Systems.* PhD thesis, Department of Electrical Engineering, University of Queensland, 1992.

[2] A.D. Back and A.C. Tsoi. FIR and IIR synapses, a new neural network architecture for time series modeling. *Neural Computation*, 3(3):375–385, 1991.

[3] Etienne Barnard, R.A. Cole, and L. Hou. Location and classification of plosive constants using expert knowledge and neural-net classifiers. *Journal of the Acoustical Society of America*, 84 Supp 1:S60, 1988.

[4] H.A. Bourlard and N. Morgan. *Connnectionist Speech Recognition: A Hybrid Approach.* Kluwer Academic Publishers, Boston, MA, 1994.

[5] B. de Vries and J.C. Principe. The Gamma model – a new neural network for temporal processing. *Neural Networks*, 5(4):565–576, 1992.

[6] J.H. Friedman. Introduction to computational learning and statistical prediction. Tutorial Presented at Neural Information Processing Systems, Denver, CO, 1995.

[7] Steve Lawrence, A.C. Tsoi, and A.D. Back. The Gamma MLP for speech phoneme recognition. In D. Touretzky, M. Mozer, and M. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 785–791. MIT Press, 1996.

[8] Kai-Fu Lee and Hsiao-Wuen Hon. Speaker-independent phone recognition using hidden Markov models. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(11):1641–1648, 1989.

[9] J.C. Principe, B. de Vries, and P. Oliveira. The gamma filter – a new class of adaptive IIR filters with restricted feedback. *IEEE Transactions on Signal Processing*, 41:649–656, 1993.

[10] J.C. Principe, J.M. Kuo, and S. Celebi. An analysis of the Gamma memory in dynamic neural networks. *IEEE Transactions on Neural Networks*, 5(2):331–337, 1994.

[11] J.J. Shynk. Adaptive IIR filtering. *IEEE ASSP Magazine*, pages 4–21, 1989.

[12] L. Yaeger, R. Lyon, and B. Webb. Effective training of a neural network character classifier for word recognition. In M.C. Mozer, M.I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, Cambridge, MA, 1997. MIT Press.

A DETERMINISTIC ANNEALING APPROACH TO DISCRIMINATIVE HIDDEN MARKOV MODEL DESIGN *

Ajit Rao, Kenneth Rose, and Allen Gersho
Department of Electrical and Computer Engineering
University of California, Santa Barbara, CA 93106.

Abstract

We present the problem of designing a classifier system based on hidden Markov models (HMMs) from a labeled training set with the objective of minimizing the rate of mis-classification. The traditional design approach divides the training set into subsets of identically labeled training vectors and independently designs the HMM corresponding to each subset of the training data using a maximum likelihood criterion. However, this approach does not achieve the minimum mis-classification objective. To design the globally optimal recognizer, all the HMMs must be jointly optimized to minimize the number of mis-classified training patterns. This is a difficult design problem which we attack using the technique of deterministic annealing (DA). In the DA approach, we introduce randomness in the classification rule and minimize the expected mis-classification rate of the random classifier while controlling the level of randomness in its decision via a constraint on the Shannon entropy. The effective cost function is smooth and converges to the mis-classification cost at the limit of zero entropy (non-random classification rule). The DA approach can be implemented via an efficient forward-backward algorithm for recomputing the model parameters.

*This work was supported in part by the National Science Foundation under grant no. NCR-9314335, the University of California MICRO program, ACT Networks, Inc., Advanced Computer Communications, Cisco Systems, Inc., DSP Group, Inc., DSP Software Engineering, Inc., Fujitsu Laboratories of America, Inc., General Electric Company, Hughes Electronics Corp., Intel Corp., Nokia Mobile Phones, Qualcomm, Inc., Rockwell International Corp., and Texas Instruments, Inc.,

0-7803-4256-9/97/$10.00 ©1997 IEEE

This algorithm significantly outperforms the standard maximum likelihood algorithm for a moderate increase in design complexity.

1 Introduction

The hidden Markov model (HMM) is commonly used as a stochastic model for time sequences. HMMs were originally applied within main-stream statistics, but the discovery of their applicability to modeling speech utterances [3, 6] has led to extensive research activity in HMMs over the last three decades. An overwhelming number of conventional speech recognition systems are based on the use of the HMM to model various speech utterances within the context of traditional discriminant-based pattern classification.

In this paper, we address the problem of recognition of time sequences modeled by HMMs. It is formally defined as the design of a recognizer based on a labeled training set (i.e., supervised learning). This problem has been extensively treated in the speech recognition literature. The most commonly used approach is to divide the training set into subsets of identically labeled training vectors and independently design HMMs for each subset of training data via maximum likelihood estimation of model parameters. After design, the system is used for recognizing new sequences through competition between the designed HMMs. The input sequence is declared to belong to the winner (the most likely model).

The starting point of our work is the realization that the above recognition problem is fundamentally a pattern classification problem. Further, the quality of the recognizer is most appropriately measured by its rate of classification error. This leads to two major observations: First, the globally optimal recognizer must be designed through *joint optimization* of all models. It is important to emphasize that the ultimate objective is not to model the sequences belonging to each class as accurately as possible, but rather, to distinguish between the classes while making as few errors as possible. As classification is performed by competition between models, it is clear that we must optimize all the model parameters simultaneously to minimize classification errors.

This also connects to the second observation, namely, that maximum likelihood is a mismatched cost for optimizing the classifier. The direct measure of success is simply the empirical rate of correct classification. It should be noted in passing that the Bayesian classifier which is optimal in the sense of minimum classification error, is a close relative of the maximum likelihood approach above. However its success depends on the availability of the precise probability distributions, including the assumption that the model structure is in complete agreement with the source. If one has only access to a reasonably short training set, the performance of maximum likelihood

may differ significantly from that of minimum classification error, as will be demonstrated in this work. We note that the shortcomings of the maximum likelihood method have been previously recognized (*e.g.* [1, 2, 4, 7]) and joint optimization approaches have been suggested.

There are several important difficulties in approaching the design problem directly, that is, by joint optimization of all model parameters so as to minimize the rate of classification error. One difficulty is that unlike maximum likelihood, this cost function is piecewise constant and all gradients with respect to parameters vanish almost everywhere (an infinitesimal change in parameter values will not change the classification of any sequence in the training set). Thus, one cannot simply use a gradient based optimization method. An important approach to address this problem appeared in [7] where the cost surface was smoothed to allow the application of gradient methods (A few weeks ago, a paper appeared [5], where this method was extended to HMM classification.). Another important difficulty is that even if the cost surface is smoothed, the optimization process tends to suffer from numerous shallow local minima that riddle this complex cost surface. Finally, one must keep in mind the difficulties associated with the computational complexity of such joint optimization.

The main contribution of this paper is a novel method for designing HMM-based recognizers. The new method is based on the deterministic annealing approach to clustering [14, 13] and in particular to its recent extension to classification [8]. By introducing randomness that is controlled by imposing the level of Shannon entropy, we obtain an effective cost function that is smooth and converges to the original classification error cost at the limit of zero entropy. Further, this process is analogous to physical annealing and hence has the capability to avoid many shallow minima that trap standard local optimization methods. It is also important to note that unlike the stochastic procedure of simulated annealing, the process here is deterministic and all randomization is taken into account by taking the expectation of the various quantities. Another important result is the development of a forward-backward algorithm (similar to Baum-Welch re-optimization) for recomputing the parameters of all models in our joint optimization framework. (Note that here we do not use maximum likelihood as our ultimate objective). This algorithm is instrumental in keeping the computational complexity manageable. The approach is shown to substantially outperform the standard maximum likelihood method at the cost of moderate increase in design complexity with respect to separate design of HMM per class.

2 The HMM classifier and its design

We address the supervised learning problem of designing a recognition system from a labeled training set, $\mathcal{T} \equiv \{(\mathbf{y}_1, c_1), (\mathbf{y}_2, c_2), ..(\mathbf{y}_N, c_N)\}$. Each *train-*

ing pattern, $\mathbf{y}_i$, is a vector of l_i observations, $\mathbf{y}_i = (\mathbf{y}_i(1), \mathbf{y}_i(2), \cdots \mathbf{y}_i(l_i))$. Further, each observation, $\mathbf{y}_i(t)$, is a discrete quantity, *i.e.* $\mathbf{y}_i(t) \in \mathcal{A} \equiv \{1, 2, \cdots, K\}$. Despite this restriction to the case of discrete observations, we note that the design methods can be easily be extended to handle continuous valued observations also. The training pattern, $\mathbf{y}_i$, belongs to class, c_i, which may be one of M classes, *i.e.* $c_i \in \mathcal{C} \equiv \{1, 2, \cdots M\}$.

The HMM recognition system consists of a set of hidden Markov models, $\{H_j,\ j = 1, 2, \cdots, M\}$, one per class index. The model, H_j has S_j states and is fully specified by the parameter set $\Lambda_j \equiv (A_j, B_j, \Pi_j)$, where following the usual convention, A_j is the $(S_j \times S_j)$ state transition probability matrix, B_j is the $(S_j \times K)$ emission probability matrix and Π_j is the (length S_j) initial state probability vector.

The classifier works as follows : Given a training pattern, $\mathbf{y}_i$, for each HMM, H_j, and for each sequence (length l_i) of states, $\mathbf{s} \equiv (s(1), s(2), \cdots, s(l_i))$ in the trellis of H_j, we determine the log likelihood, $l(\mathbf{y}_i, \mathbf{s}, H_j)$, that the observation $\mathbf{y}_i$ is generated via the state sequence, $\mathbf{s}$. Hence,

$$l(\mathbf{y}_i, \mathbf{s}, H_j) = \log \Pi_j(s(1)) + \sum_{t=1}^{l_i - 1} \log A_j(s(t), s(t+1)) + \sum_{t=1}^{l_i} \log B_j(s(t), \mathbf{y}_i(t)). \tag{1}$$

Here, $A_j(m, n)$ is the (m,n) element of the matrix, A_j. Similarly, $B_j(m, k)$ is the (m, k) element of matrix, B_j, and $\Pi_j(m)$ is the mth component of the vector, Π_j.

Next, we maximize the log likelihood over all state sequences in the trellis of HMM, H_j, and determine

$$d_j(\mathbf{y}_i) = \max_{\mathbf{s} \in \mathcal{S}_{l_i}(H_j)} l(\mathbf{y}_i, \mathbf{s}, H_j). \tag{2}$$

Here, $\mathcal{S}_l(H_j)$ is the set of all state sequences of length l in the trellis of HMM, H_j. The quantity, $d_j(\mathbf{y}_i)$ thus represents the log likelihood of the state sequence in model H_j, that most likely generated $\mathbf{y}_i$. Interpreting $d_j(\cdot)$ as the discriminant for class j, we adopt the traditional discriminant-based classification approach to define the classifier operation as :

$$C(\mathbf{y}_i) = arg \max_j d_j(\mathbf{y}_i). \tag{3}$$

We refer to this definition as the "best path" discriminant [1]. This classification system can be viewed as a competition between paths. The observation is ultimately labeled by the class index of the HMM to which the winning path belongs. One advantage of the "best path" discriminant classifier is that the search for the most likely path (choosing a state sequence, $\mathbf{s}$, that maximizes (2)) can be reduced to a sequential optimization problem that can be solved via an efficient dynamic programming algorithm (Viterbi search).

[1] Our design method can be easily modified to the case where the discriminant is obtained by appropriate averaging of the likelihood over all paths in the class model.

2.1 HMM classifier design

The problem of HMM classifier design can be stated as the joint optimization of the HMM parameters, $\{\Lambda_j\}$, to minimize the empirical probability of misclassification measured over the training set,

$$\min_{\{\Lambda_j\}} P_e = 1 - \frac{1}{N} \sum_{i=1}^{N} \delta(C(\mathbf{y}_i), c_i) \tag{4}$$

where δ is the error indication function: $\delta(u, v) = 1$ if $u = v$ and 0 otherwise.

The most important difficulty in this optimization is that the cost, P_e, is a piecewise constant function of the optimization variables. As a result, we cannot use traditional gradient descent based optimization methods - the gradients are zero almost everywhere. One approach [7] to circumvent this difficulty is to replace the piecewise cost function by a smooth approximation to it. While the modified cost function is amenable to descent-based optimization, in practice, there are numerous shallow local minima on the complex cost surface that can easily trap optimization methods based on simple descent. In the next section, we present a novel approach based on deterministic annealing to simultaneously tackle the piecewise nature of the cost function and the problem of shallow local minima traps.

3 Deterministic Annealing approach

We take as our starting point, the deterministic annealing approach to clustering, vector quantization [14] and related optimization problems [13] and its extension to structurally-constrained clustering problems [8]. The extended method can handle problems involving structural constraints on the clustering rule *e.g.* tree structured vector quantization, pattern classifiers based on parametric discriminant functions etc. We have recently applied the extended DA method successfully to the design of standard pattern classifiers [8], regression functions [9, 12, 11] and source coding systems [10]. The work presented in this paper represents an important extension of the method to handle time sequences that are modeled by HMMs.

We cast the optimization problem within a probabilistic framework and maintain that, during design, it is useful to consider a randomized HMM classifier system. In the randomized classifier, given an observation, a winning state sequence is randomly chosen from among all state sequences in all the HMMs. This (random) choice of the winning state sequence is based on a probability distribution - we replace the best-path discriminant rule which associates a pattern to a unique winning state sequence by a randomized best-path discriminant rule that associates each pattern, $\mathbf{y}_i$, to every state sequence, $\mathbf{s}$, in the trellis of every model, H_j, with a proba-

bility, $P(\mathbf{y}_i, \mathbf{s}, H_j)$. Naturally, these probabilities are normalized such that $\sum_j \sum_{\mathbf{s} \in \mathcal{S}_{l_i}(H_j)} P(\mathbf{y}_i, \mathbf{s}, H_j) = 1$.

The probabilities, $P(\mathbf{y}_i, \mathbf{s}, H_j)$, are obtained in a systematic manner: We first note that the non-random best-path discriminant rule may be expressed as minimization over $\mathbf{s}_i \in \bigcup_j \mathcal{S}_{l_i}(H_j)$ of the cost function,

$$D = \frac{1}{N} \sum_i l(\mathbf{y}_i, \mathbf{s}_i, H_j). \tag{5}$$

After randomness is introduced, this cost function is replaced by the expected cost,

$$< D >= \frac{1}{N} \sum_i \sum_j \sum_{\mathbf{s} \in \mathcal{S}_{l_i}(H_j)} P(\mathbf{y}_i, \mathbf{s}, H_j) l(\mathbf{y}_i, \mathbf{s}, H_j), \tag{6}$$

which is minimized, while simultaneously enforcing a level of randomness though a constraint on the Shannon entropy,

$$H = -\frac{1}{N} \sum_i \sum_j \sum_{\mathbf{s} \in \mathcal{S}_{l_i}(H_j)} P(\mathbf{y}_i, \mathbf{s}, H_j) \log P(\mathbf{y}_i, \mathbf{s}, H_j). \tag{7}$$

In particular, we optimize $< D >$ subject to $H = \hat{H}$. The probability distribution obtained via this constrained optimization problem is the Gibbs distribution,

$$P(\mathbf{y}_i, \mathbf{s}, H_j) = \frac{e^{\gamma l(\mathbf{y}_i, \mathbf{s}, H_j)}}{\sum_{j'} \sum_{\mathbf{s}' \in \mathcal{S}_{l_i}(H_{j'})} e^{\gamma l(\mathbf{y}_i, \mathbf{s}', H_{j'})}}. \tag{8}$$

The value of Shannon entropy, $\hat{H}$, corresponding to this Gibbs distribution is determined by the positive *scale parameter*, γ. This parameter also controls the "randomness" of the distribution. For $\gamma = 0$, the distribution over paths is uniform. For finite, positive values of γ, the Gibbs distribution indicates that we assign higher probabilities of winning to state sequences with higher log likelihoods. In the limiting case of $\gamma \to \infty$, the random classification rule reverts to the non-random "best path" classifier, which assigns a non-zero probability of winning only to the path with the highest log likelihood as in (2).

The random classifier's expected rate of misclassification (over the training set) can be calculated as

$$< P_e >= 1 - \frac{1}{N} \sum_{i=1}^{N} \sum_{\mathbf{s} \in \mathcal{S}_{l_i}(H_{c_i})} P(\mathbf{y}_i, \mathbf{s}, H_j) \tag{9}$$

Next, we pose the problem of optimizing this random HMM classifier (choosing $\{\Lambda_j\}$ and γ) to minimize the expected mis-classification probability of (9). However, simply minimizing (9) over all Gibbs distributions chooses one that is non-random ($\gamma \rightarrow \infty$). While such a non-random, best-path classifier is the eventual goal of this design method, we wish to enforce the "non-randomness" gradually during the optimization, to avoid shallow local minima traps.

As such, we follow the philosophy underlying the deterministic annealing approach and pose the problem of minimizing $< P_e >$ while maintaining a level of randomness in the classifier through a constraint on the entropy, $H = \hat{H}$. This constrained optimization problem is equivalently, the minimization of the unconstrained Lagrangian cost function,

$$\min_{\{\Lambda_j\},\gamma} L \equiv < P_e > -TH, \tag{10}$$

where T is the Lagrange parameter that we refer to as the "temperature" because of an interesting analogy in statistical physics.

3.1 Analogy to statistical physics

The Lagrangian minimization of (10) reminds us of the definition of thermal equilibrium in statistical physics. The quantity, L, is analogous to the Helmholtz free energy of a thermodynamic system with average energy $< P_e >$, entropy over energy states, H and temperature, T. This free energy is the quantity that is minimized when this thermodynamic system is at thermal equilibrium at temperature, T.

From the optimization viewpoint, we are particularly interested in thermal equilibrium at $T = 0$ which corresponds to direct minimization of $< P_e >$, our ultimate objective. The analogy to physical systems suggests that to minimize $< P_e >$, it is useful to implement an annealing process, that is, gradually lower the temperature while maintaining the system at thermal equilibrium. We start with a very high value of T, where the sole objective is entropy maximization, which is achievable by the uniform distribution. Reducing T gradually from this high value, we repeat the process of minimizing L until $T = 0$, where the sole objective is optimizing $\{\Lambda_j\}$ and γ to minimize P_e.

After this annealing process, we also include as a final step, a "quenching" mechanism - we optimize $\{\Lambda_j\}$ to minimize P_e, while increasing γ from its optimal value at $T = 0$, in gradual steps, to a very high value. When γ is sufficiently high, the classifier reduces to the non-random "best-path" classifier.

The annealing process yields a sequence of solutions at decreasing levels of entropy and P_e leading to the "best-path" classifier in the limit. The DA method is not a stochastic method like simulated annealing, but instead based

on the optimization of the deterministically computed expectation, L, at each temperature. This minimization is achieved by a series of gradient descent steps with the following expressions for the gradients :

$$\frac{\partial L}{\partial \Lambda_j} = \frac{1}{N} \sum_i \sum_{\mathbf{s} \in \mathcal{S}_{l_i}(H_j)} L(\mathbf{y}_i, \mathbf{s}, H_j) P(\mathbf{y}_i, \mathbf{s}, H_j) \{ \frac{\partial l(\mathbf{y}_i, \mathbf{s}, H_j)}{\partial \Lambda_j} - < \frac{\partial l(\mathbf{y}_i, \mathbf{s}, H_j)}{\partial \Lambda_j} >_j \}$$

and

$$\frac{\partial L}{\partial \gamma} = \frac{1}{N} \sum_i \sum_j \sum_{\mathbf{s} \in \mathcal{S}_{l_i}(H_j)} L(\mathbf{y}_i, \mathbf{s}, H_j) P(\mathbf{y}_i, \mathbf{s}, H_j) \{ l(\mathbf{y}_i, \mathbf{s}, H_j) - < l(\mathbf{y}_i, \mathbf{s}, H_j) > \}$$

Here, $L(\mathbf{y}_i, \mathbf{s}, H_j) = T\gamma l(\mathbf{y}_i, \mathbf{s}, H_j) - \delta(j, c_i)$. The operation, $< f(\cdot) >_j$, represents an expectation of the (state-sequence dependent) $f(\cdot)$ function over the state sequences in the trellis of HMM, H_j. Hence,

$$< \frac{\partial l(\mathbf{y}_i, \mathbf{s}, H_j)}{\partial \Lambda_j} >_j = \sum_{\mathbf{s} \in \mathcal{S}_{l_i}(H_j)} P(\mathbf{y}_i, \mathbf{s}, H_j) \frac{\partial l(\mathbf{y}_i, \mathbf{s}, H_j)}{\partial \Lambda_j} \qquad (11)$$

Similarly, $< f(\cdot) >$ represents the expectation of the $f(\cdot)$ function over all state sequences in the trellises of all the HMMs. Hence,

$$< l(\mathbf{y}_i, \mathbf{s}, H_j) > = \sum_j \sum_{\mathbf{s} \in \mathcal{S}_{l_i}(H_j)} P(\mathbf{y}_i, \mathbf{s}, H_j) l(\mathbf{y}_i, \mathbf{s}, H_j). \qquad (12)$$

An important aspect of the proposed method is the discovery of an efficient forward-backward algorithm to determine these gradient parameters. Note that the summations in the gradient expressions are over all state sequences in the trellis of HMMs. The number of paths depends exponentially on the number of states in the HMM. However, these summations can be efficiently computed via a forward-backward algorithm which reduces the number of computations substantially (proportional to square of the number of states in the HMM) thus cutting down on computational complexity and memory requirements. The complexity of the DA method scales similarly to the maximum likelihood method with respect to the number of states and training vectors.

4 Experimental Results

We have performed preliminary simulations to determine the usefulness of our new design method. We experimented on designing simple (2,3 and 4

class) classifier systems for eight different data sets of 2000 vectors each. Allowing three to six states in each Markov model, we designed HMM classifier systems using the maximum likelihood and deterministic annealing methods. We observe that the proposed DA approach improved the classification performance consistently and considerably. Over the experiment's data sets, the rate of misclassification was reduced by factors of 1.2 to 3. Table 1 details the results.

We are currently investigating the effectiveness of the design method on real-world speech data to demonstrate its advantages for the speech recognition problem.

Dataset	1	2	3	4
No. of Classes	2	2	2	3
P_e (ML)	17.4%	31.6%	26.5%	28.7%
P_e (DA)	6.5%	21.7%	18.7%	20.9%
Dataset	5	6	7	8
No. of Classes	3	3	3	4
P_e (ML)	27.0 %	32.5%	24.9 %	42.3 %
P_e (DA)	21.0%	27.3%	17.4%	31.7%

Table 1: A comparison of the mis-classification rates obtained for HMM classifiers designed from eight classified training sets of 2000 patterns each. Each set consists of data from 2,3 or 4 classes. ML represents a Max. likelihood design algorithm and DA represents the deterministic annealing algorithm.

5 Conclusion

In this paper we propose a novel training method for HMM classifier systems that jointly optimizes all the models to minimize the true cost, namely, the rate of mis-classification. At the cost of moderate increase in complexity, considerable improvements in recognition rates are obtained.

References

[1] L.R. Bahl, P.F. Brown, P.V. DeSouza, R.L. Mercer, "A new algorithm for the estimation of hidden Markov model parameters", Proc. ICASSP-88, pp. 493-496.

[2] L.R. Bahl, P.F. Brown, P.V. DeSouza, R.L. Mercer, "Maximum Mutual Information estimation of hidden Markov model parameters", Proc. ICASSP-86, pp. 49-52, Tokyo, Japan.

[3] J.K. Baker, "Stochastic modeling for automatic speech recognition", In D.R.Reddy, ed., Speech Recognition, New York: Academic Press, pp. 521-542, 1975.

[4] H. Boulard, Y. Konig, N. Morgan, "REMAP: Recursive Estimation and Maximization of A Posteriori Probabilities - Application to Transition-Based Connectionist Speech Recognition", ICSI technical Report TR94-064, Internat, Computer Science Inst. CA.

[5] H. Watanabe, S. Katagiri, "HMM speech recognizer based on discriminative metric design", Proc. of ICASSP, 1997, vol. 4, pp. 3237-3240.

[6] F. Jelinek, "Continuous speech recognition by statistical methods", Proceedings of the IEEE, vol. 64, pp. 532-556, Apr. 1972.

[7] B.H. Juang, and S. Katagiri, "Discriminative learning for minimum error classification", IEEE Trans. on Sig. Proc., vol. 40, pp 3043-3054, 1992.

[8] D. Miller, A. Rao, K. Rose, A. Gersho, "A Global Optimization Technique for Statistical Classifier Design", IEEE Transactions on Sig. Proc., vol. 44, pp 3108-3122, 1996.

[9] A. Rao, D. Miller, K. Rose, A. Gersho, " An Information-theoretic Approach for Statistical Regression with Model Growth by Bifurcations", Computing Science and Statistics, Proc. of the 27th Symposium on the Interface, 1995, pp 220 - 224.

[10] A. Rao, D. Miller, K. Rose, A. Gersho, " A Generalized VQ Method for Combined Compression and Estimation", Proc. of the IEEE ICASSP, 1996, pp. 2032 - 2035.

[11] A. Rao, D. Miller, K. Rose, A. Gersho, " A Deterministic Annealing Approach for Parsimonious Design of Piecewise Regression Models ", submitted for publication.

[12] A. Rao, D. Miller, K. Rose, A. Gersho, "Mixture of Experts Regression Modeling by Deterministic Annealing", accepted for publication in the IEEE Transactions on Signal Processing, Nov. 1997.

[13] K. Rose, E. Gurewitz, G.C. Fox, "Constrained clustering as an optimization method", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 15, pp. 785-794, 1993.

[14] K. Rose, E. Gurewitz, G.C. Fox, " Vector quantization by deterministic annealing ", IEEE Trans. on Inform. Theory, vol. 38, pp 1249-1258, 1992.

An Improved Training Algorithm for Support Vector Machines

Edgar Osuna
C.B.C.L.
MIT E25-201
Cambridge, MA, 02139
eosuna@ai.mit.edu
tel: (617) 252 1723

Robert Freund
O.R. Center
MIT E40-149A
Cambridge, MA, 02139
rfreund@mit.edu
tel: (617) 253 8997

Federico Girosi
C.B.C.L.
MIT E25-201
Cambridge, MA, 02139
girosi@ai.mit.edu
tel: (617) 253 0548

Abstract

We investigate the problem of training a Support Vector Machine (SVM) [1, 2, 7] on a very large date base (e.g. 50,000 data points) in the case in which the number of support vectors is also very large (e.g. 40,000). Training a SVM is equivalent to solving a linearly constrained quadratic programming (QP) problem in a number of variables equal to the number of data points. This optimization problem is known to be challenging when the number of data points exceeds few thousands. In previous work, done by us as well as by other researchers, the strategy used to solve the large scale QP problem takes advantage of the fact that the expected number of support vectors is small ($< 3,000$). Therefore, the existing algorithms cannot deal with more than a few thousand support vectors. In this paper we present a decomposition algorithm that is guaranteed to solve the QP problem and that does not make assumptions on the expected number of support vectors. In order to present the feasibility of our approach we consider a foreign exchange rate time series data base with 110,000 data points that generates 100,000 support vectors.

1 Introduction

In this paper we consider the problem of training a Support Vector Machine (SVM), a pattern classification algorithm recently developed by V. Vapnik and his team at AT&T Bell Labs. [1, 2, 7]. SVM can be seen as a new way to train polynomial, neural network, or Radial Basis Functions classifiers, based on the idea of *structural risk minimization* rather than *empirical risk* minimization [1]. From the implementation point of view, training a SVM

[1] The name of SVM is due to the fact that one of the outcomes of the algorithm, in addition to the parameters for the classifier, is a set of data points (the "support vectors") which contain, in a sense, all the "relevant" information about the problem.

0-7803-4256-9/97/$10.00 ©1997 IEEE

is equivalent to solving a linearly constrained Quadratic Programming (QP) problem in a number of variables equal to the number of data points. This problem is challenging when the size of the data set becomes larger than a few thousands, which is often the case in practical applications. A number of techniques for SVM training have been proposed [7, 4, 5, 6]. However, many of these strategies take advantage of the following assumptions, or expectations: **1)** The number of support vectors is small, with respect to the the number of data points; **2)** the total number of support vectors does not exceed a few thousands (e.g. < 3.000). Since the ratio between the number of support vectors and the total number of data points (averaged over the probability distribution of the input variables) is an upper bound on the generalization error, the previous assumptions are violated in the following cases: **1)** the problem is "difficult", so that the generalization error will be large and therefore the proportion of support vectors is high, or **2)** the data set is so large (say 300,000) that even if the problem can have small generalization error (say 1%) the number of support vectors will be large (in this case around 3,000).
The algorithm that we present in this paper does not make the above mentioned assumptions. It should be noticed, however, that in the case in which the assumptions above are satisfied the algorithm *does* take advantage of them. The algorithm is similar in spirit to the algorithm that we proposed in [4] (that was limited to deal with few thousands support vectors): it is a decomposition algorithm, in which the original QP problem is replaced by a sequence of smaller problems that is proved to converge to the global optimum. Although the experiments we report in this paper concern a classification problem, the current algorithm can also be used, with minimal modifications, to train the new version of the SVM, that can deal with regression as well as classification.
The plan of the paper is as follows: in the next section we briefly sketch the ideas underlying SVM. Then in section 3 we present our new algorithm, in section 4 we show some results of our implementation on a financial data set with 110,000 data-points with as many as 100,000 support vectors and in section 5 we summarize the paper.

2 Support Vector Machines

In this section we briefly sketch the SVM algorithm and its motivation. A more detailed description of SVM can be found in [7] (chapter 5) and [2].
We start from the simple case of two linearly separable classes. We assume that we have a data set $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{\ell}$ of labeled examples, where $y_i \in \{-1, 1\}$, and we wish to select, among the infinite number of linear classifiers

that separate the data, one that minimizes the generalization error, or at least an upper bound on it (this is the idea underlying the structural risk minimization principle [7]). V. Vapnik showed [7] that the hyperplane with this property is the one that leaves the maximum margin between the two classes [1], where the margin is defined as the sum of the distances of the hyperplane from the closest point of the two classes.
If the two classes are non-separable the SVM looks for the hyperplane that maximizes the margin and that, at the same time, minimizes a quantity proportional to the number of misclassification errors. The trade off between margin and misclassification error is controlled by a positive constant C that has to be chosen beforehand. In this case it can be shown that the solution to this problem is a linear classifier $f(\mathbf{x}) = \text{sign}(\sum_{i=1}^{\ell} \lambda_i y_i \mathbf{x}^T \mathbf{x}_i + b)$ whose coefficients λ_i are the solution of a QP problem, defined over the hypercube $[0, C]^\ell$, whose precise statement will be given in section 3 (see eq. 1). Since the quadratic form is minimized in the hypercube $[0, C]^\ell$, the solution will have a number of coefficients λ_i exactly equal to zero. Since there is a coefficient λ_i associated to each data point, only the data points corresponding to non-zero λ_i (the "support vectors") will influence the solution. Intuitively, the support vectors are the data points that lie at the border between the two classes. It is then clear that a small number of support vectors indicates that the two classes can be well separated.
This technique can be extended to allow for non-linear decision surfaces. This is done by projecting the original set of variables $\mathbf{x}$ in a higher dimensional *feature space*: $\mathbf{x} \in R^d \Rightarrow \mathbf{z}(\mathbf{x}) \equiv (\phi_1(\mathbf{x}), \ldots, \phi_n(\mathbf{x})) \in R^n$ (where n is possibly infinite) and by formulating the linear classification problem in the feature space. Vapnik proves that there are certain choices of features ϕ_i for which the solution has the following form:

$$f(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{\ell} \lambda_i y_i K(\mathbf{x}, \mathbf{x}_i) + b\right)$$

where $K(\mathbf{x}, \mathbf{y})$ is a symmetric positive definite kernel function that depends on the choice of the features and represent the scalar product in the feature space. In table (1) we list some choices of the kernel function proposed by Vapnik: notice how they lead to well known classifiers, whose decision surfaces are known to have good approximation properties.

3 Training a Support Vector Machine

In this section we present a decomposition algorithm that, without making assumptions on the expected number of support vectors, allows us to train a

Kernel Function	Type of Classifier
$K(\mathbf{x}, \mathbf{x}_i) = \exp(-\|\mathbf{x} - \mathbf{x}_i\|^2)$	Gaussian RBF
$K(\mathbf{x}, \mathbf{x}_i) = (\mathbf{x}^T \mathbf{x}_i + 1)^d$	Polynomial of degree d
$K(\mathbf{x}, \mathbf{x}_i) = \tanh(\mathbf{x}^T \mathbf{x}_i - \Theta)$	Multi Layer Perceptron

Table 1: Some possible kernel functions and the type of decision surface they define.

SVM on a large data set by solving a sequence of smaller QP problems. The two key issues to be considered are:

1. **Optimality Conditions**: These conditions allow us to decide computationally whether the problem has been solved optimally at a particular iteration of the original problem. Section 3.1 states and proves optimality conditions for the QP given by (1).

2. **Strategy for Improvement**: If a particular solution is not optimal, this strategy defines a way to improve the cost function and is frequently associated with variables that violate optimality conditions. This strategy will be stated in section 3.2.

Using the results of sections 3.1 and 3.2 we will then formulate our decomposition algorithm in section 3.3.

3.1 Optimality Conditions

The QP problem that we have to solve in order to train a SVM is the following [1, 2, 7]:

$$\begin{array}{lll} \underset{\Lambda}{\text{Minimize}} & W(\mathbf{\Lambda}) = -\mathbf{\Lambda}^T \mathbf{1} + \frac{1}{2}\mathbf{\Lambda}^T D \mathbf{\Lambda} & \\ \text{subject to} & & \\ & \mathbf{\Lambda}^T \mathbf{y} = 0 & (\mu) \\ & \mathbf{\Lambda} - C\mathbf{1} \leq \mathbf{0} & (\mathbf{\Upsilon}) \\ & -\mathbf{\Lambda} \leq \mathbf{0} & (\mathbf{\Pi}) \end{array} \tag{1}$$

where $(\mathbf{1})_i = 1$, $D_{ij} = y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$, μ, $\mathbf{\Upsilon}^T = (\upsilon_1, \ldots, \upsilon_\ell)$ and $\mathbf{\Pi}^T = (\pi_1, \ldots, \pi_\ell)$ are the associated Kuhn-Tucker multipliers. The choice of the kernel K is left to the user, and it depends on the decision surfaces one expects to work best. Since D is a positive semi-definite matrix (the kernel function K is positive definite), and the constraints in (1) are linear, the Kuhn-Tucker, (KT) conditions are necessary and sufficient for optimality. The KT conditions are as follows:

$$
\begin{array}{ll}
\nabla W(\mathbf{\Lambda}) + \mathbf{\Upsilon} - \mathbf{\Pi} + \mu\mathbf{y} & = \mathbf{0} \\
\mathbf{\Upsilon}^T(\mathbf{\Lambda} - C\mathbf{1}) & = 0 \\
\mathbf{\Pi}^T\mathbf{\Lambda} & = 0 \\
 & \\
\mathbf{\Upsilon} & \geq \mathbf{0} \\
\mathbf{\Pi} & \geq \mathbf{0} \\
 & \\
\mathbf{\Lambda}^T\mathbf{y} & = 0 \\
\mathbf{\Lambda} - C\mathbf{1} & \leq \mathbf{0} \\
-\mathbf{\Lambda} & \leq \mathbf{0}
\end{array}
\tag{2}
$$

In order to derive further algebraic expressions from the optimality conditions (2), we assume the existence of some λ_i such that $0 < \lambda_i < C$, and consider the three possible values that each component of $\mathbf{\Lambda}$ can have:

1. **Case:** $0 < \lambda_i < C$
   From the first three equations of the KT conditions we have:

$$(D\mathbf{\Lambda})_i - 1 + \mu y_i = 0 \tag{3}$$

   Using the results in [2] and [7] one can show that this implies that $\mu = b$.

2. **Case:** $\lambda_i = C$
   From the first three equations of the KT conditions we have:

$$(D\mathbf{\Lambda})_i - 1 + \upsilon_i + \mu y_i = 0 \tag{4}$$

   It is useful to define the following quantity:

$$g(\mathbf{x}_i) = \sum_{j=1}^{\ell} \lambda_j y_j K(\mathbf{x}_i, \mathbf{x}_j) + b \tag{5}$$

   Using the fact that $\mu = b$ and requiring the KT multiplier υ_i to be positive one can show that the following conditions should hold:

$$y_i g(\mathbf{x}_i) \leq 1 \tag{6}$$

3. **Case:** $\lambda_i = 0$
   From the first three equations of the KT conditions we have:

$$(D\mathbf{\Lambda})_i - 1 - \pi_i + \mu y_i = 0 \tag{7}$$

By applying a similar algebraic manipulation as the one described for case 2, we obtain

$$y_i g(\mathbf{x}_i) \geq 1 \tag{8}$$

3.2 Strategy for Improvement

The optimality conditions derived in the previous section are essential in order to devise a decomposition strategy that guarantees that at every iteration the objective function is improved. In order to accomplish this goal we partition the index set in two sets B and N, where the set B is called the *working set.* Then we decompose $\mathbf{\Lambda}$ in two vectors $\mathbf{\Lambda}_B$ and $\mathbf{\Lambda}_N$, keeping fixed $\mathbf{\Lambda}_N$ and allowing changes only in $\mathbf{\Lambda}_B$, thus defining the following subproblem:

$$\begin{array}{lll} \text{Minimize} & W(\mathbf{\Lambda}_B) & = -\mathbf{\Lambda}_B^T\mathbf{1} + \frac{1}{2}\left[\mathbf{\Lambda}_B^T D_{BB}\mathbf{\Lambda}_B + \mathbf{\Lambda}_B^T D_{BN}\mathbf{\Lambda}_N + \right. \\ & & \left. + \mathbf{\Lambda}_N^T D_{NB}\mathbf{\Lambda}_B + \mathbf{\Lambda}_N^T D_{NN}\mathbf{\Lambda}_N\right] - \mathbf{\Lambda}_N^T\mathbf{1} \\ \mathbf{\Lambda}_B & & \\ \text{subject to} & & \\ & \mathbf{\Lambda}_B^T\mathbf{y}_B + \mathbf{\Lambda}_N^T\mathbf{y}_N & = 0 \\ & \mathbf{\Lambda}_B - C\mathbf{1} & \leq \mathbf{0} \\ & -\mathbf{\Lambda}_B & \leq \mathbf{0} \end{array} \tag{9}$$

where $(\mathbf{1})_i = 1$, $D_{\alpha\beta}$ is such that $D_{ij} = y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$, with $i \in \alpha, j \in \beta$, and C is a positive constant. Using this decomposition we notice that:

- The terms $-\mathbf{\Lambda}_N^T\mathbf{1} + \frac{1}{2}\mathbf{\Lambda}_N^T D_{NN}\mathbf{\Lambda}_N$ are constant within the defined subproblem.
- Since $K(\mathbf{x}, \mathbf{y})$ is a symmetric kernel, the computation of $\mathbf{\Lambda}_B^T D_{BN}\mathbf{\Lambda}_N + \mathbf{\Lambda}_N^T D_{NB}\mathbf{\Lambda}_B$ can be replaced by $2\mathbf{\Lambda}_B^T\mathbf{q}_{BN}$, where:

$$(\mathbf{q}_{BN})_i = y_i \sum_{j \in N} \lambda_j y_j K(\mathbf{x}_i, \mathbf{x}_j) \qquad i \in B \tag{10}$$

  This is a very important simplification, since it allows us to keep the size of the subproblem independent of the number of fixed variables $\mathbf{\Lambda}_N$, which translates into keeping it also independent of the number of support vectors.

- We can replace *any* λ_i, $i \in B$, with *any* λ_j, $j \in N$ (i.e. there is no restriction on their value), without changing the cost function or the feasibility of both the subproblem and the original problem.

- If the subproblem is optimal before such a replacement, the new subproblem is optimal if and only if λ_j satisfies the Optimality Conditions for the appropriate case (3 cases described above).

The previous statements lead to the following more formal propositions:

Proposition 3.1 *("Build down"): moving a variable from B to N leaves the cost function unchanged, and the solution is feasible in the subproblem.*

Proof: Let $B' = B \setminus \{k\}$ and $N' = N \cup \{k\}$. Then:

$$\begin{aligned}
W(\mathbf{\Lambda}_B, \mathbf{\Lambda}_N) &= -\sum_{i\in B}\lambda_i - \sum_{i\in N}\lambda_i + \frac{1}{2}\left[\sum_{i,j\in B}\lambda_i\lambda_j D_{ij} + 2\sum_{i\in B}\lambda_i\sum_{j\in N}\lambda_j D_{ij} + \right. \\
&\quad \left. + \sum_{i,j\in N}\lambda_i\lambda_j D_{ij}\right] \\
&= -\sum_{i\in B'}\lambda_i - \lambda_k - \sum_{i\in N}\lambda_i + \frac{1}{2}\left[\sum_{i,j\in B'}\lambda_i\lambda_j D_{ij} + 2\lambda_k\sum_{i\in B'}\lambda_i D_{ik} + \right. \\
&\quad \left. + 2\lambda_k\sum_{j\in N}\lambda_j D_{jk} + 2\sum_{i\in B'}\lambda_i\sum_{j\in N}\lambda_j D_{ij} + \lambda_k^2 D_{kk} + \sum_{i,j\in N}\lambda_i\lambda_j D_{ij}\right] \\
&= -\sum_{i\in B'}\lambda_i - \sum_{i\in N'}\lambda_i + \frac{1}{2}\left[\sum_{i,j\in B'}\lambda_i\lambda_j D_{ij} + 2\sum_{i\in B'}\lambda_i\sum_{j\in N'}\lambda_j D_{ij} + \right. \\
&\quad \left. + \sum_{i,j\in N'}\lambda_i\lambda_j D_{ij}\right] \\
&= W(\mathbf{\Lambda}_{B'}, \mathbf{\Lambda}_{N'})
\end{aligned}$$

The solution $(\mathbf{\Lambda}_{B'}, \mathbf{\Lambda}_{N'})$ is feasible in the subproblem since:

$$\begin{aligned}
0 &= \mathbf{\Lambda}_B^T\mathbf{y}_B + \mathbf{\Lambda}_N^T\mathbf{y}_N \\
&= \mathbf{\Lambda}_{B'}^T\mathbf{y}_{B'} + \lambda_k y_k + \mathbf{\Lambda}_N^T\mathbf{y}_N \\
&= \mathbf{\Lambda}_{B'}^T\mathbf{y}_{B'} + \mathbf{\Lambda}_{N'}^T\mathbf{y}_{N'}
\end{aligned}$$

and the bound constraints are always unaffected.

Proposition 3.2 *("Build up"): moving a variable that violates the optimality conditions from N to B gives a strict improvement in the cost function when the subproblem is re-optimized.*
prop:propdown

Proof: This is a direct consequence of Proposition 3.1 and the fact that Kuhn-Tucker conditions are necessary and sufficient for optimality.

3.3 The Decomposition Algorithm

Using the results of the previous sections we are now ready to formulate our decomposition algorithm:

1. Arbitrarily choose $|B|$ points from the data set.
2. Solve the subproblem defined by the variables in B.
3. While there exists some $j \in N$, such that:
   - $\lambda_j = 0$ and $g(\mathbf{x}_j)y_j < 1$
   - $\lambda_j = C$ and $g(\mathbf{x}_j)y_j > 1$
   - $0 < \lambda_j < C$ and $g(\mathbf{x}_j)y_j \neq 1$,

   replace any λ_i, $i \in B$, with λ_j and solve the new subproblem given by:

$$\begin{array}{lll} \text{Minimize} & W(\mathbf{\Lambda}_B) & = -\mathbf{\Lambda}_B^T\mathbf{1} + \frac{1}{2}\mathbf{\Lambda}_B^T D_{BB}\mathbf{\Lambda}_B + \mathbf{\Lambda}_B^T\mathbf{q}_{BN} \\ \mathbf{\Lambda}_B & & \\ \text{subject to} & & \\ & \mathbf{\Lambda}_B^T\mathbf{y}_B + \mathbf{\Lambda}_N^T\mathbf{y}_N & = 0 \\ & \mathbf{\Lambda}_B - C\mathbf{1} & \leq \mathbf{0} \\ & -\mathbf{\Lambda}_B & \leq \mathbf{0} \end{array} \tag{11}$$

where:

$$(\mathbf{q}_{BN})_i = y_i \sum_{j \in N} \lambda_j y_j K(\mathbf{x}_i, \mathbf{x}_j) \qquad i \in B \tag{12}$$

Notice that we have omitted the constant term $-\mathbf{\Lambda}_N^T\mathbf{1} + \frac{1}{2}\mathbf{\Lambda}_N^T D_{NN}\mathbf{\Lambda}_N$ in the cost function, and that according to Proposition 3.2, this algorithm will strictly improve the objective function at each iteration and therefore will not cycle. Since the objective function is bounded ($W(\mathbf{\Lambda})$ is convex quadratic and the feasible region is bounded), the algorithm must converge to the global optimal solution in a finite number of iterations.

4 Implementation and Results

We have implemented the decomposition algorithm using MINOS 5.4 [3] as the solver of the sub-problems. We tested our technique on a problem known for being "difficult": a foreign exchange rate time series that was used in the 1992 Santa Fe Institute Time Series Competition, in which we looked at the sign of the change of the time series, rather than its value. We considered data sets of increasing sizes, up to 110,000 points, obtaining up to 100,000 support vectors. Figure 4 shows the relationship between training times, number of data points and number of support vectors in our experiments. The training time on a SUN Sparc 20 with 128 Mb of RAM ranged from 3 hours for 10,000 support vectors to 48 hours for 40,000 support vectors. The results that we obtain are comparable to the results reported in [8] using a Neural Networks approach, where generalization errors around 53% were reported. The purpose of this experiment was not to benchmark SVM's on this specific problem, but to show that its use in a problem with as many as 100,000 support vectors is computationally tractable.

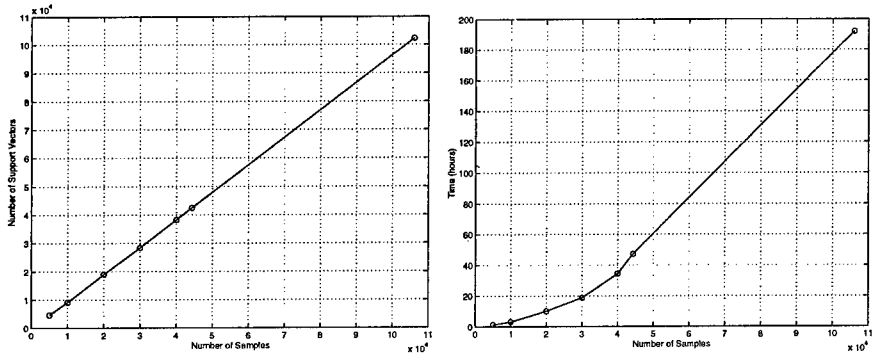


Figure 1: (a)Number of support vectors Vs. number of data points. (b) Training time Vs. number of data points.

5 Summary and Conclusions

In this paper we have presented a novel decomposition algorithm that can be used to train Support Vector Machines on large data sets that contain a large number of support vectors. The current version of the algorithm has been tested with a data set of 110,000 data points and 100,000 support

vectors on a machine with 40 Mb of RAM. No attempts to optimize and speed up the algorithm have been made yet. We believe that this algorithm starts to meet the increasing need to deal with data sets where both the number of data points and the number of support vectors are of the order of 10^5. Problems with these characteristics are likely to be found in the area of financial markets, where lots of data maybe available but little generalization error is expected.

Acknowledgements

The authors would like to thank Sayan Mukherjee for his help in collecting and preprocessing the data.

References

[1] B.E. Boser, I.M. Guyon, and V.N. Vapnik. A training algorithm for optimal margin classifier. In *Proc. 5th ACM Workshop on Computational Learning Theory*, pages 144–152, Pittsburgh, PA, July 1992.

[2] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:1–25, 1995.

[3] B. Murtagh and M. Saunders. Large-scale linearly constrained optimization. *Mathematical Programming*, 14:41–72, 1978.

[4] E. Osuna, R. Freund, and F. Girosi. Support vector machines: Training and applications. A.I. Memo 1602, MIT A. I. Lab., 1997.

[5] M. Schmidt. Identifying speakers with support vectors networks. In *Proceedings of Interface '96*, Sydney, July 1996.

[6] B. Schölkopf, C. Burges, and V. Vapnik. Extracting support data for a given task. In U.M. Fayyad and R. Uthurusamy, editors, *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, Menlo Park, CA, 1995. AAAI Press.

[7] V. Vapnik. *The Nature of Statistical Learning Theory.* Springer, New York, 1995.

[8] X. Zhang. Non-linear predictive models for intra-day foreign exchange trading. Technical report, PHZ Partners, August 1993.

Classification with Linear Networks Using an On-line Constrained LDA Algorithm

Jose C. Principe, Dongxin Xu

Computational NeuroEngineering Laboratory
EB #33, Electrical and Computer Engineering Department
University of Florida, Gainesville, FL 32611, USA
{principe,xu}@cnel.ufl.edu

Abstract

Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) are two statistical tools utilized in many signal processing areas such as data compression and pattern recognition. On-line algorithms such as Oja's rule have found wide application for PCA and it has been generalized in our previous paper to LDA (Fisher criterion) using the framework of gradient descent learning. In this paper, the rule is further extended to accept the case of constraints in the feature extraction stage as is often necessary for real world applications. As examples, the new rule is applied to regularizers in the form of both a 2-dimensional (2-D) Gaussian filter for hand-written digit classification and determination of the memory depth of the Gamma filter for isolated word recognition. Results show the good behavior of the learning rule and the advantage of using regularization for improved generalization.

1. Introduction

There are basically two different roles for feature extraction depending upon the nature of the signal processing problem. In signal representation PCA has been shown to be the best linear feature extractor, while LDA using the Fisher criterion is a common choice in classification. In both cases, the feature extraction stage is crucial for the overall performance. Recently, alternate methods based on information theory have been proposed which seek statistical independent features also known as independent component analysis - ICA ([4]). Although naive compared to ICA, PCA is well established and on-line training methods such as Oja's rule have been proved very robust. More recently, several on-line training algorithms have been independently proposed for LDA, [1], [2], [3], but the state-of-the-art is less developed. In our previous paper [1], Oja's rule was generalized to LDA under the framework of the gradient descent method, which we called generalized Oja's rule (GOR). The experiments in [1] have shown the effectiveness of the proposed GOR.

The purpose of this paper is two-fold: First, GOR will be further extended to the

0-7803-4256-9/97/$10.00

case where constraints are imposed on the linear feature extractor. This is an important practical problem, because preprocessing can be thought of as a form of constraint and in this way the function of the preprocessor is integrated with the classifier for better performance. One application is regularization or smoothing which is needed to improve generalization and can be formulated as constraints on the model weights. Based on the extended GOR, we derive as special cases the training rule for LDA applied to digit recognition regularized with a 2-D Gaussian filter, and to the training of the memory depth of the Gamma memory for word recognition. The second purpose of the paper is to compare the performance of different classifiers based on PCA, non-regularized and regularized LDA, and a MLP (multilayer perceptron) on hand-written digit data. The adaptation of the Gamma memory recursive parameter suffers in general from local minima and in particular from a lack of an adequate criterion in temporal pattern recognition settings. We show that the time scale parameter is much more robustly adjusted by the GOR than by the usual back-propagation method.

2. Generalized Oja's Rule with Constraints

This work addresses the linear network shown in Figure 1 which has an output $y = w^T x$. In [1], we showed that the generalized Oja's rule updates the weights as:

$$\begin{cases} \Delta w = \sum_i y_i (x_i - y_i b) \\ w^T b = 1 \end{cases} \tag{1}$$

where b is a vector which serves as the basic measurement for the problem under analysis. For PCA the measurement vector is w, the weight vector itself. For LDA the measurement vector is the class mean difference m, which depends upon the scatter matrix [1].

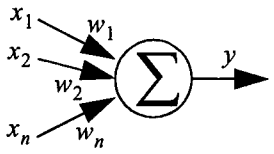


Figure 1. Network for PCA or 2 class LDA

We formulate a constraint as a functional dependence of the weights w on a vector of parameters

$$w(v) = (w_1(v), w_2(v), \ldots, w_n(v))^T \tag{2}$$

where $v \in R^m$ are parameters and usually $m < n$. We assume that w is differen-

tiable with respect to v. There are two main reasons to use this formulation. First, we can use the chain rule to compute the sensitivity of the cost with respect to the new parameter set. Secondly, linear filters with kernels can be regarded as constraint on is projection weights.

Let $y = f(w, x) = w(v)^T x$ denotes the output of the linear network when the input is x. By applying the gradient method, we get (3) which is the extended GOR for the case with constraints:

$$\begin{cases} \Delta v = \Delta w^T \cdot \dfrac{\partial w}{\partial v} = \sum_i y_i \left(\dfrac{\partial}{\partial v} f(w, x) - y_i \dfrac{\partial}{\partial v} f(w, b) \right) \\ f(w, b) = 1 \end{cases} \tag{3}$$

If the constraints can be implemented in a recursive form such as in the Gamma filter [6], the calculation of the partial derivative in (3) can be done by backpropagation, as will be shown in the following.

3. Regularized LDA with a 2-D Gaussian Filter

Real world problems are very often data bound, i.e. in high dimensional spaces there are no enough data to train the model and to avoid overfitting. One alternative is to use regularization or smoothing. As an example, let's look at hand-written digit recognition problem, where each digit is a binary or multi-level 2-D image of size $P \times Q$ (here 24×18). A quadratic classifier requires the estimation on the order of 93,528 parameters which is impractical. Using the analytic approach at least 10 times these samples are recommended by practitioners. Iterative techniques are able to solve the problem with less data, but overfitting becomes a problem. LDA using the Fisher criterion also requires the estimation of the covariance matrix, so this reasoning applies to our method.

To prevent overfitting, a smoothing constraint is applied to the projection vector w. One possible solution is to apply a linear combination of 2-D Gaussian kernels to the input image which is equivalent to a 2-D linear lowpass filter with an impulse response which is Gaussian. The advantage of using a Gaussian filter is that each Gaussian kernel is symmetric and the problem can be easily formulated in the scheme of equation (3). Note that the Gaussian kernel is used as a weighting function over the input instead of as a nonlinear operator applied to the input space as done in the radial basis networks (RBFs).

In Fig.2, each pixel in the image is indexed by (I, J) $(I= 1, \ldots, P; J= 1, \ldots, Q)$ and is ordered to form the data vector x. Correspondingly, the elements of the projection vector w can also be indexed by (I, J) and ordered in the same way. Fig. 2

depicts the 2-D Gaussian kernels on a 2-D grid. The center of the top left Gaussian kernel is (I_0, J_0) , where I_0 and J_0 are real values, not necessarily integers.

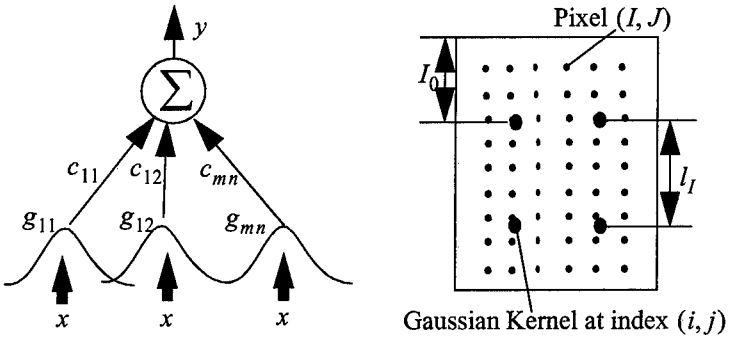


Figure 2. Regularized LDA Network with Gaussian kernel

Thus, the center indexed by (i,j) is determined by $((i-1)l_I + I_0, (j-1)l_J + J_0)$, where l_I and l_J are also real values and represent the distance between neighboring Gaussian kernels in the vertical and horizontal direction respectively. So, the output of the network can be expressed as:

$$y = w^T x, \qquad w = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij} g_{ij}$$

$$g_{ij}(I,J) = \exp\left(-\frac{(I-(i-1)l_I-I_0)^2 + (J-(j-1)l_J-J_0)^2}{\sigma}\right) \tag{4}$$

where g_{ij} represents the Gaussian kernel at index (i,j) and c_{ij} is the corresponding linear combination coefficient. σ is a positive real value related to the variance of the Gaussian kernel and can be used to adjust the degree of smoothing.

Let $A \bullet B$ represent the element by element multiplication between two matrices. Let $D(i) = I_0 + (i-1)l_I - I$ and $D(j) = J_0 + (j-1)l_J - J$ be the vectors ordered in the same way as x and w. By applying (3), we can derive the adaptation rule for c_{ij}, I_0, l_I and σ as follows:

$$\Delta c_{ij} = \sum y(x - yb)^T g_{ij}$$

$$\Delta I_0 = \left(\sum y(x-yb)^T\right)\left(\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij} g_{ij} \bullet D(i)\right)\left(-\frac{2}{\sigma}\right)$$

$$\Delta l_I = \left(\sum y(x-yb)^T\right)\left(\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij} g_{ij} \bullet D(i)(i-1)\right)\left(-\frac{2}{\sigma}\right)$$

$$\Delta\sigma = \left(\sum y(x - yb)^T\right)\left(\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}g_{ij} \bullet (D(i)^2 + D(j)^2)\right)/\sigma^2$$

J_0, J_J have similar formulas to I_0 and I_I respectively. Notice that with this formalism the regularization constant σ can be adapted for optimal performance in the training set.

4. Experiment on Hand-written Digit Recognition

The data includes 100 hand-written digit "3" and 100 hand-written digit "8" with size 24×18. We choose digits "3" and "8" because they are similar, differing mainly in their left side. We select 10 exemplars of each for training to illustrate an extreme case of a small data set. Solving LDA by the numerical method would not be possible because of the scatter matrix singularity. On-line PCA, non-regularized LDA and Gaussian regularized LDA are applied to the data, trained by Oja's rule, GOR and the extended GOR respectively. A multilayer perceptron (MLP) is also applied to the data for comparison, although all the above methods are linear networks. The results are shown in Figures 3, 4, 5, and 6. All the left images represent the normalized projection (largest eigenvector for PCA and Fisher direction for LDA) after convergence, with white pixels corresponding to positive values. All the histograms at the right reflect the distribution of digits "3" and "8" after the projection. The smallest classification errors are also indicated in the figures.

From Fig. 3, we can recognize both the "3" and "8", which suggests that PCA preserves the important information for representing both the "3" and "8". However, the two classes overlap in the PCA projection even for the training data set, which simply means that PCA is not the most appropriate feature extractor for classification. Fig. 4 and Fig 5 show the results for both the conventional and Gaussian regularized LDA. Fig. 6 shows the result for a MLP with 5 hidden and one output nodes applied directly to the data. We can see that the training data of the two classes have been separated completely by all three methods, and regularized LDA obtains the best generalization on the test set for this problem. The left images simply show that non-regularized LDA is too noisy and overfitted to the training data, while regularized LDA still sustains a degree of smoothness and preserves separation.

Notice the asymmetric and long tailed distributions created by the MLP outputs, which are characteristic of a nonlinear mapper. Recall also that the LDA networks are linear, but the weights have been determined to enhance separation. Although our network is not a RBF network, we believe that GOR could be used advantageously to train the linear weights in RBFs.

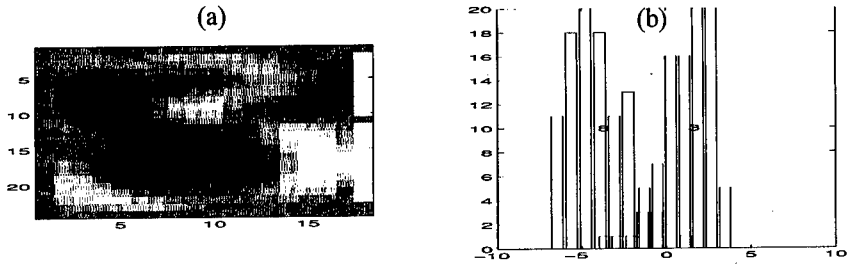


Figure 3. Result of PCA. (a) weight image; (b) data distribution (training)

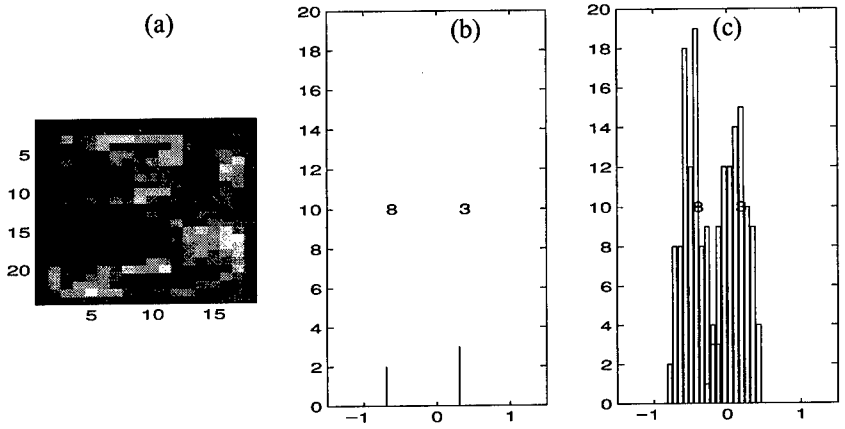


Figure 4. Result of Non-regularized LDA. 7 errors for testing data.
(a) weight image; (b) training data distribution; (c) testing data distribution

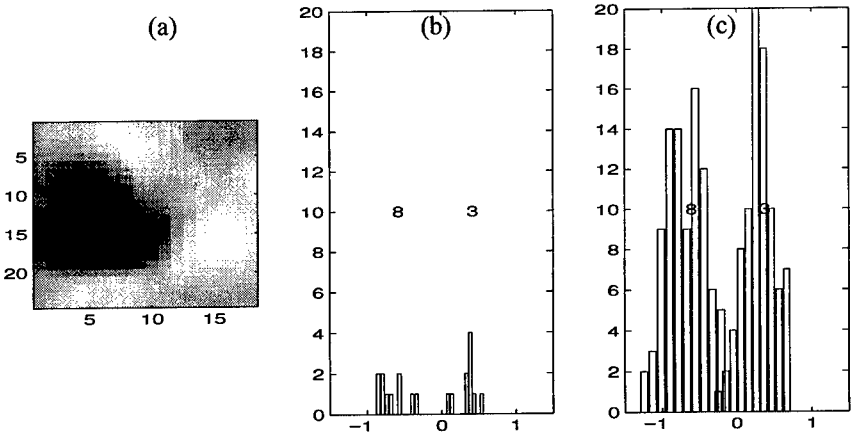


Figure 5. Result of Regularized LDA. 3 errors for testing data.
(a) weight image; (b) training data distribution; (c) testing data distribution

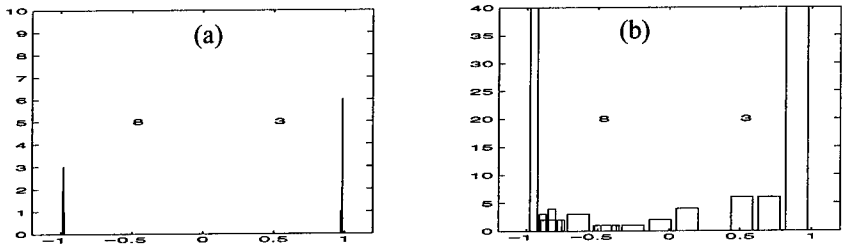


Figure 6. Result of Multilayer Perceptron (5 hidden nodes). 6 errors for testing data. (a) training data distribution; (b) testing data distribution

5. Regularized LDA applied to Gamma memory training

The gamma memory is the building block for both the gamma filter and the gamma neural model ([6], [7]). The hallmark of this structure is a delay element which is recursive and can be adapted on-line with the output error using gradient descent. For signal representation this adaptation method is reasonable, but for classification of temporal patterns the method is a compromise when the classes require different memory depths. In this section, the recursive parameter of the gamma memory and the weights of the gamma filter (Figure 7) will be adapted using regularized LDA. This structure constitutes the first layer of the focused gamma network [7].

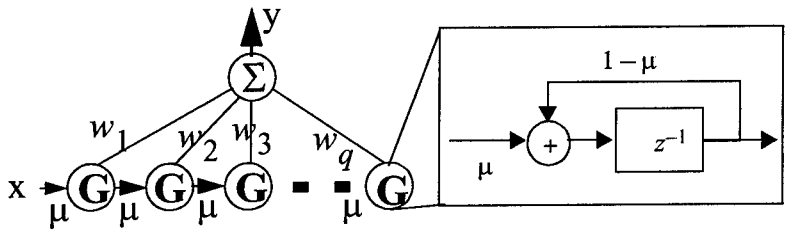


Figure 7. Gamma Memory + Linear Classifier (Gamma Filter).

As shown in Figure 7, data are first projected onto the gamma memory and then a linear projection is applied. The model can be formalized by (5)

$$y = f(x) = w^T G^T x = (Gw)^T x = \left((g_1, g_2, \ldots g_q) \begin{bmatrix} w_1 \\ \ldots \\ w_q \end{bmatrix} \right)^T x \qquad (5)$$

where q is the number of taps in the Gamma memory, $g_i \in R^n$ is a column vector which is the order-reversed Gamma kernel (impulse response) at tap i truncated with length n. G is the function of Gamma parameter μ. By applying the generalized Oja's rule, the adaptation rule for the model can be obtained as follows:

$$\begin{cases} \Delta w = \sum_i y_i \left(\frac{\partial}{\partial w} f(x_i) - y_i \frac{\partial}{\partial w} f(b) \right) = \sum_i y_i (G^T x_i - y_i G^T b) \\ \Delta \mu = \sum_i y_i \left(\frac{\partial}{\partial \mu} f(x_i) - y_i \frac{\partial}{\partial \mu} f(b) \right) \\ \text{Normalization: scaling } w \text{ so that } f(b) = w^T G^T b = 1 \end{cases} \quad (6)$$

where $G^T x$ and $G^T b$ are the projection of input x and base vector b to the gamma memory respectively. Since the gamma memory is a recursive structure, $\partial f(x_i)/\partial \mu$ and $\partial f(b)/\partial \mu$ can be calculated by backpropagation through time (BPTT). As (5) and Fig. 7 show, the model can be regarded as a filter. So, Gw is the truncated order-reversed impulse response with length n and can be generated by activating the model with an impulse. The normalization can be done at the same time by adjusting the weights $w_1, \ldots, w_q$.

It is not difficult to extend the result of (6) to the case of independent multi-channel Gamma memories for each channel as required for speech recognition. If we regard the whole projection as the concatenation of the projection of each channel, then the base vector for PCA in this case is the concatenation of the truncated order-reversed impulse response for each channel. For the two-class Fisher LDA, the base vector is the class mean difference which is easy to understand and obtain.

6. Experiment on Isolated Word Recognition

The data are two classes isolated spoken words "wash" and "cash" from the TIMIT database. There are only 7 examples for each word in the database. For this problem, we are solely interested in finding the optimum value of the gamma parameter μ that best discriminates between the two words. The use of long tap delay lines is discourage due to the large networks that they produce, so the gamma memory is appealing and has produced good results [8]. Common sense tells us that the major difference between "wash" and "cash" lies in the beginning of the words. Different values of the gamma parameter will let the model focus on different regions of the input pattern. Therefore, finding an optimum μ such that the model concentrate on the most discriminant part of the input data will improve performance. Since back-propagation with output MSE (mean squared error) can also be used to adjust the gamma parameter in the gamma filter, this method will be compared with the GOR training.

A 16-channel constant Q filter bank (Mel scale) is used as the pre-processing of the

speech signal. For a clear illustration, only the results of 1-channel data are presented (15th channel). The top panel of Figure 8 was obtained with the optimal weights and by stepping μ from 0 to 2. This panel measures discrimination (Fisher criterion) as a function of μ and clearly shows that the ability to discriminate between the two words depends heavily upon the value selected (from 0.3 to 0.5; values of μ larger than 1 are expected to produce low discrimination (filter becomes an highpass filter)). We can see that the optimal μ is around 0.4 which is consistent with the observation that long delays are needed. Longer delays will compromise resolution too much due to the lowpass nature of the kernel, and discrimination drops. When the backpropagation method is utilized (second panel) the value of μ obtained is far from the optimum. This attest the difficulty of adapting μ for pattern recognition applications due to the conflicting requirements of finding the best scale to represent more than one class. The generalized Oja's rule (third panel) converges to the optimal value of μ within 100 iterations.

7. Conclusion

This paper shows that the generalized Oja's rule can be extended even when the weights are a function of hidden variables as may be necessary to solve practical problems. LDA is a linear procedure that exploits the difference among classes contained in both the mean and the covariance (or scatter) matrix. So it is sensitive to poor estimations of parameters which are most often caused by insufficient data. Regularization prevents the model from overfitting even when the data yields rank deficient covariance matrices. This is one obvious advantage of our on-line LDA over the conventional (numeric) LDA implementation where a matrix inverse is required. The idea of optimally smoothing the data before LDA can be effectively incorporated in the extended GOR learning rule as was demonstrated here. Note that the parameters of the Gaussian smoother are adapted during operation, so the method extracts as much information as possible from the training data. Although the examples are all two-class problems, GOR can be extended to multiple classes as we will report in a following paper.

The results for the adaptation of the recursive parameter in the gamma memory are also important since they present for the first time a technique to adapt the μ parameter for temporal pattern recognition applications where the method of minimizing the output mean square error is not the most appropriate.

Acknowledgments: This work was partially supported by NSF grant ECS-9510715.

REFERENCES

[1] Principe J., Xu D., and Wang C., "Generalized Oja's Rule for Linear Discriminant Analysis with Fisher Criterion", Proc. ICASSP'97

[2] Chatterjee C. and Roychowdhury V. "Self-Origination with Hetero-Associative Networks for Linear Discriminant Analysis", World Congress on Neural Networks, San Diego, California, Sept., 1996.
[3] Diamantaras K. and Kung S., "Principal Component Neural Networks, Theory and Applications", John Wiley & Sons, Inc, New York, 1996
[4] Deco, G., "An Information-Theoretic Approach to Neural Computing" New York, Springer, 1996
[5] Kapur J., Kesavan H., "Entropy Optimization Principles with Application" Academic Press Inc. 1992
[6] de Vries, B. and J. C. Principe, "The gamma model - a new neural model for temporal processing," Neural Networks, vol. 5, no. 4, pp. 565-576, 1992.
[7] Principe J., Kuo J-M, Celebi S., "An analysis of the Gamma memory in dynamic neural networks," IEEE Trans. NN, vol 5, #2, 331, March 1994.
[8] Tracy J., and Principe J., "Isolated word recognition with the gamma model", J. of Art. Neural Nets, vol 1, #4, 481, 1994.

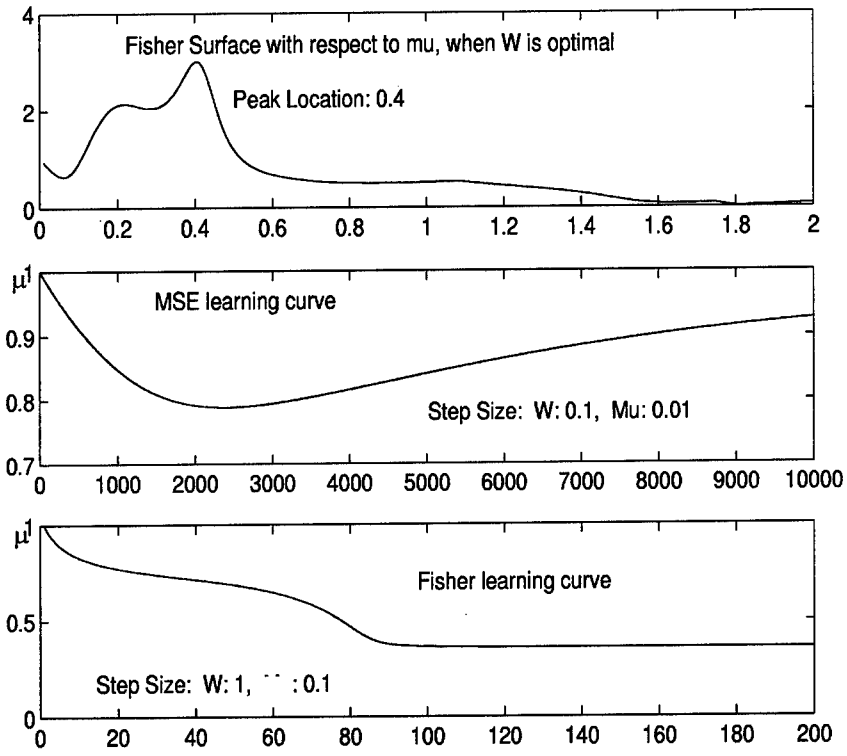


Figure 8. Adaptation of μ First panel shows that the best value to discriminate is μ=0.4. Second panel shows that MSE drives μ towards 0.9. Third panel shows that GOR finds the right value of μ in around 100 iteration.

Combination of Adaptive Signal Processing and Neural Classification Using an Extended Backpropagation Algorithm *

A. Doering, H. Witte
Institute of Medical Statistics, Computer Science and Documentation
Friedrich Schiller University Jena D–07740 Jena, Jahnstrasse 3, Germany
{doering, iew}@imsid.uni-jena.de

Abstract

Beside the use of purely neural systems, the combination of preprocessing units and neural classifiers has been used for a variety of signal segmentation and classification tasks. Whereas this approach reduces the input dimensionality as well as the complexity of the classification problem, its performance crucially depends on a proper preprocessing scheme, i.e., feature extraction. In this contribution, adaptive preprocessing units (frequency–selective quadrature filters) are proposed that can be adjusted in order to provide optimal features. The mean frequencies of the filters are tuned to minimize the classification error. Both FIR– and IIR–based filters are introduced and compared with respect to their convergence properties and the classification results. Results for the solution of an EEG segmentation task using the combined system are given.

1 Introduction

A common problem in biosignal processing is the classification (i.e., labeling of samples) and segmentation (i.e., discrimination between samples) of signals. From a more general viewpoint, this problem states a supervised learning task with the ordering of the learning samples given by time. Due to their well–known universal approximation capability, Neural Networks (NNs)

*This work was supported by the the German Ministry for Education and Research (project "Clinical Oriented Neurosciences", 01 ZZ 9602) and the Thuringian Ministry for Science, Research and Arts (project ITHERA, B511-95004).

0-7803-4256-9/97/$10.00 ©1997 IEEE

seem to be a suited approach for a solution. Whereas a variety of recurrent NN models have been proposed, particularly for prediction tasks, we focus in the sequel on feedforward networks mainly for three reasons:

1. Efficient, robust (with respect to the initial conditions) and easily implementable training algorithms that do not have to consider stability restrictions are available for feedforward NNs.

2. Recurrency is known to "blur" segment borders. Whereas in the linear case (IIR filter) this influence is relatively easy to be estimated, this is generally not possible for nonlinear systems.

3. A consistent theory for the generalization of recurrent networks has not yet been developed. However, there is some evidence that at least partially recurrent networks might lack intrinsic generalization capability, e.g. with respect to sampling rate changes [8].

Using feedforward networks, the relevant part of the signal's past has to be fed explicitly into the network. Back and Tsoi [2] proposed the use of FIR–synapses that correspond to the parallel presentation of delayed samples. One can avoid the significant increase of the input dimensionality connected herewith, if a suited preprocessing of the signal is feasible that extracts the information for the classification task. A number of authors have adopted this combination of signal processing units with neural classification e.g. [7], [5]. If, however the parameters of the preprocessing system are not controlled by the classification error, the performance of the combined system strongly depends on the a priori available knowledge, thus questioning the very advantage of the neural approach as a modelfree method.

In the following, we introduce LTI filters with adaptable mean frequencies that constitute the basis of a more general class of nonlinear adaptive signal processing units (ASPUs). An algorithm for the adaptation of these systems will be given in 3. In section 4 the proposed approach is applied to the segmentation of discontinuous EEG.

2 Adaptive Signal Processing Units

A signal processing unit provides a mapping of the time series $\{x(n)\}_{n=1,2,\ldots}$ onto a set of f features

$$z_i(n) = \Phi_i(\mathbf{p}_i;\ x(n),\ x(n-1),\ \ldots) \quad i = 1 \ldots f$$

which are fed into a neural classifier with the input–output–relation $o(n) = o(\mathbf{z}(n), \mathbf{W})$ (with $\mathbf{W}$ denoting the weights). In order to integrate the determination of the optimal parameter vector $\mathbf{p}_i^*$ that controls the calculation of the features into the NN training, the gradient $\nabla_{\mathbf{p}_i} z_i(n)$ has to be computed.

2.1 FIR bandpass filter with adaptable mean frequency

The desired frequency response

$$H_d(\omega) = \begin{cases} 1 & \omega_m - \Delta\omega/2 \leq |\omega| \leq \omega_m + \Delta\omega/2 \\ 0 & \text{otherwise} \end{cases}$$

can be approximated using the FIR coefficients

$$h_k = \begin{cases} \frac{2\nu_k}{\pi(k-\frac{M-1}{2})} \cos(\omega_m(k-\frac{M-1}{2})) \sin(\frac{\Delta\omega}{2}(k-\frac{M-1}{2})) & \\ \frac{\Delta\omega}{\pi}\nu_k & \text{if } k = \frac{M-1}{2} \end{cases}$$

$$z(n) = \sum_{k=0}^{M-1} h_k x(n-k)$$

$(k = 0 \ldots M-1)$ where the ν_k correspond to a suited window function. The computation of the gradient is equivalent to a filtering of the input time series with the coefficients h'_k

$$h'_k = -\frac{2}{\pi}\nu_k \sin(\omega_m(k-\frac{M-1}{2})) \sin(\frac{\Delta\omega}{2}(k-\frac{M-1}{2}))$$

$$\frac{\partial z(n)}{\partial \omega_m} = \sum_{k=0}^{M-1} h'_k x(n-k)$$

As $\mathbf{h}$ and $\mathbf{h}'$ are symmetric, both the bandpass and gradient filter realize a linear phase. Despite its computational ease, the FIR filter suffers from several drawbacks:

1. The minimal bandwith depends on the filter degree, $\Delta\omega \geq \frac{4\pi}{M}$.
2. Due to its rippled amplitude gain, the adaptation of the mean frequency is rather sensitive against local minima. (For harmonic input signals it can be shown that for arbitrary error functions $e(n) = e\,(y(n), o\,(z(n), \mathbf{W}))$ the gradient

$$\sum_{n=1}^{\infty} \frac{\partial e(n)}{\partial \omega_m} = \sum_{n=1}^{\infty} \frac{\partial e(n)}{\partial z(n)} \frac{\partial z(n)}{\partial \omega_m}$$

has at least as many zeros as there are sidelobes in the frequency response.)

As particularly the last point seems to be a severe limitation for an adaptive preprocessing system, we introduce the adaptable IIR resonance filter.

2.2 IIR resonance filter with adaptable mean frequency

An IIR system with the poles $re^{\pm j\omega_0}$ realizes the frequency response

$$H(\omega) = \frac{b_0}{(1 - re^{j(\omega-\omega_0)})(1 - re^{j(\omega+\omega_0)})}. \quad (1)$$

As the poles are conjugate complex, this system is realizable:

$$\begin{aligned} z(n) + a_1 z(n-1) + a_2 z(n-2) &= b_0 x(n) \qquad (2) \\ a_1 &= -2r\cos(\omega_0) \\ a_2 &= r^2 \\ b_0 &= (1-r)\sqrt{1+r^2-2r\cos(\omega_0)} \end{aligned}$$

For poles near the unit circle (r clearly has to be smaller than 1 to ensure stability) the mean (or resonance) frequency is approximately equal to ω_0. The amplitude gain is unimodal and the $3dB$–bandwith does not primarily depend on the order:

$$\Delta\omega_{3db} \approx \frac{1-r}{\sqrt{r}} \quad \text{for 2nd order systems}$$

(although an increase of order corresponds to a concatenation of systems of 2nd order and thus naturally decreases the bandwith). For the off–line adaptation the computation of the gradient is simple:

$$\begin{aligned} \frac{\partial z(n)}{\partial \omega_0} &= \frac{\partial b_0}{\partial \omega_0} x(n) - \sum_{i=1}^{2} \frac{\partial a_i}{\partial \omega_0} z(n-i) - \sum_{i=1}^{2} a_i \frac{\partial z(n-i)}{\partial \omega_0} \qquad (3) \\ &= \frac{2(1-r)\sin(2\omega_0)}{\sqrt{1+r^2-2r\cos(\omega_0)}} x(n) - 2r\sin(\omega_0) z(n-1) + \\ &\quad 2r\cos\omega_0 \frac{\partial z(n-1)}{\partial \omega_0} - r^2 \frac{\partial z(n-2)}{\partial \omega_0} \end{aligned}$$

Eq. (3) can also be used for the on–line adaptation, if the changes of the mean frequency are small during any time interval $(t, t+\tau)$, where τ denotes the group delay around the mean frequency

$$\tau \approx \frac{2r^2}{1-r^2} \quad r \geq 0.8.$$

Note that as their recursive coefficients are identical, the stability of the resonance filter implies the stability of the gradient filter. An extension of (2) and (3) onto higher order filters is straightforward.

The IIR filter seems to be superior for adaptation as it eliminates both mentioned problems of the FIR filter. It does however have a nonlinear phase gain resulting in a frequency dependent group delay. This effect can be reduced using a suited allpass filter [3].

2.3 Frequency–selective Quadrature Filters with adaptable mean frequency

Frequency–selective Quadrature Filters (FSQFs) can be constructed as a combination of bandpass filters described above and generic broad–band QFs,

that can be approximated using FIR filters of order M_{QF} with coefficients $\mathbf{q}$ [6]. Thus one gets

$$z(n) = \sum_{k=0}^{M_{\mathrm{QF}}} q_k z'(n-k)$$

$$\frac{\partial z'(n)}{\partial \omega_m} = \sum_{k=0}^{M_{\mathrm{QF}}} q_k \frac{\partial z'(n)}{\partial \omega_m}$$

where z' denotes the output of either an IIR or FIR bandpass filter.

These filters can be used to detect the envelope of an amplitude–modulated signal given a mixture of superposed components or noise. The adaptation task could for example be to adapt the mean frequency of the FSQF to the (unknown) frequency of the carrier of an AM–signal, assuming that the modulating function is known.

3 Extended Backpropagation for Training ASPUs and NNs

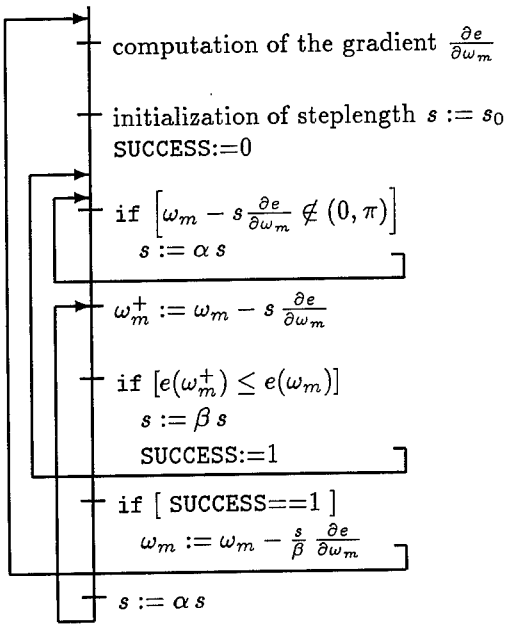


Figure 1: Algorithm for controlling the steplength (parameter s). $\alpha < 1$, $\beta > 1$

The systems proposed in the previous section enable us to compute the gradient of the output error (usually the squared difference of the target and the NN's output) with respect to the network weights (standard backpropagation) and the mean frequencies. However, as the values for ω_m, ω_0 are restricted on the intervall $(0, \pi)$, the usual gradient descent technique can not be applied. For the deterministic off–line adaptation a variety of methods for the solution of restricted optimization problems can be used. Since most of them transform the task into a series of unrestricted problems, they can not be extended onto the on–line adaptation. A simple, yet effective method that can be applied to both approaches is to control the step length in order to guarantee that none of the new parameters leaves the admissible range (fig. 1).

4 Segmentation of discontinuous EEG using Neural Networks

4.1 Data Material

The discontinuous EEG of newborns ($\approx$ 27th to 32nd week of conceptual age) is dominated by the altering occurrence of burst and interburst patterns. In [1] it is pointed out that a pattern–selective analysis is needed for an efficient quantification of the functional development of the brain in healthy newborns as well as of the severity of disturbances of newborns at risk. For the separation of burst and interburst it is essential to detect the "initial wave" at the beginning of the burst with an interindividually different frequency of about 2.8...12.8 Hz and to track the significantly increased broad band power in the frequency range 0.8...16 Hz.

For the training as well as for the classification records of frontal electrodes (Fp_1, Fp_2) of healthy newborns between the 27th and the 31st week (conceptual age) sampled at 128 Hz have been used. The records have been segmented into bursts and interbursts by a medical expert.

4.2 Preprocessing on the basis of FIR filters

The underlying assumption for the use of FSQFs as a preprocessing method is that the occurrence of a burst is characterized by the consecutive emerging of power maxima in different frequency bands. This assumption has been verified by different studies [9], [1], [4]. Thus a burst can in principle be detected using a combination of amplitude demodulators working in different frequency ranges.

Due to the application of adaptable FSQFs, the frequency ranges can be chosen in order to optimize the classification results. Table 1 shows the initial and the adapted values of the mean frequencies of a bank of FSQFs based on FIR filters ($M = 101$) which was coupled with a (4–3–1) MLP with sigmoidal node functions. Both the first and the second band converge to a

band no.	f_m (initial) [Hz]	Δf (fixed) [Hz]	f_m (after training) [Hz]
1	5.5	1	5.8
2	7.9	1	5.8
3	10.2	5	11.1
4	11.8	7	13.0

Table 1: initial values and values after training for the mean frequencies of the QF–bank (FIR–based)

band no.	f_m (initial) [Hz]	Δf_{3dB} (fixed) [Hz]	f_m (after training) [Hz]
1	10	0.2	14
2	16	0.2	5.7
3	22	0.2	23

Table 2: initial values and values after training for the mean frequencies of the QF–bank (IIR–based)

frequency that matches the assumed value of the "initial wave" (fig. 2). The classification results are shown in fig. 3.

4.3 Preprocessing on the basis of IIR filters

In order to demonstrate its robustness, the start values for the QF–bank based on IIR–filters have been chosen with greater intervalls and in greater distance to the frequency of the "initial wave" (tab. 2). Figure 2 (B) shows the convergence of the respective mean frequencies. The results of the IIR filter validate the assumption that the frequency range around 5.8 Hz contains information which is essential for the segmentation of burst and interburst patterns. Band #3 seems to be less important, as its mean frequency is little affected by the adaptation procedure. Band #2 is covered by both band #3 and band #4 of the FIR–based system.

Figure 3 shows a segment of the original signal and the corresponding classification (a (3–3–2) MLP was chosen). The IIR approach performs slightly better, particularly concerning the detection of the burst onset.

5 Conclusions

A general scheme for the integration of adaptable preprocessing units into the training process of a neural classifier has been introduced. In contrast to a "pure neural" approach this method allows the implementation of problem–specific knowledge. In contrast to static preprocessing, however, the initial choice of the preprocessing system needs not to be perfect as its parameters can be adapted during a process very similar to network training.

The use of both FIR and IIR filters for the construction of frequency

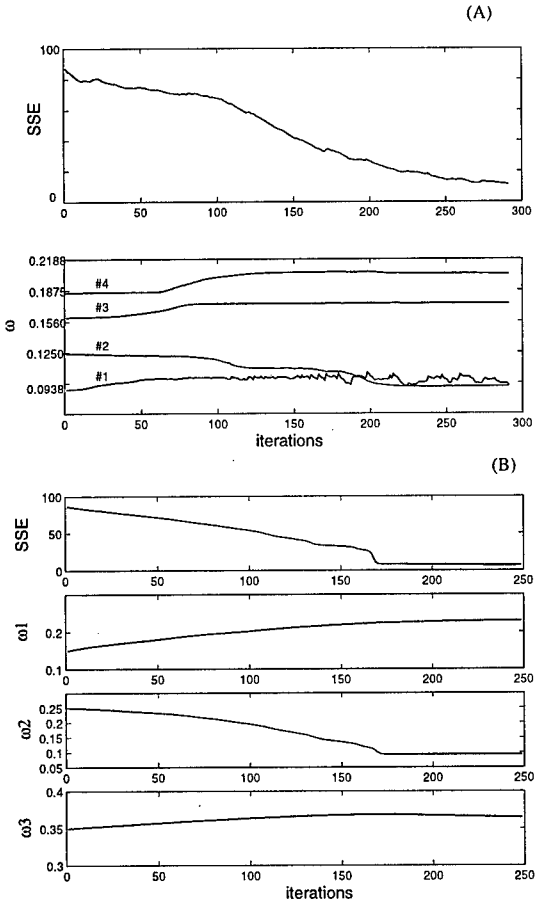


Figure 2: Evolution of the training error (SSE) versus the convergence of the mean frequencies $(\omega_1, \omega_2, \omega_3)$ for the FIR filter (A) and the IIR filter (B) based FSQFs. The ω_i are given in parts of the Nyquist frequency, i.e. $f_i = \omega_i \times 64Hz$

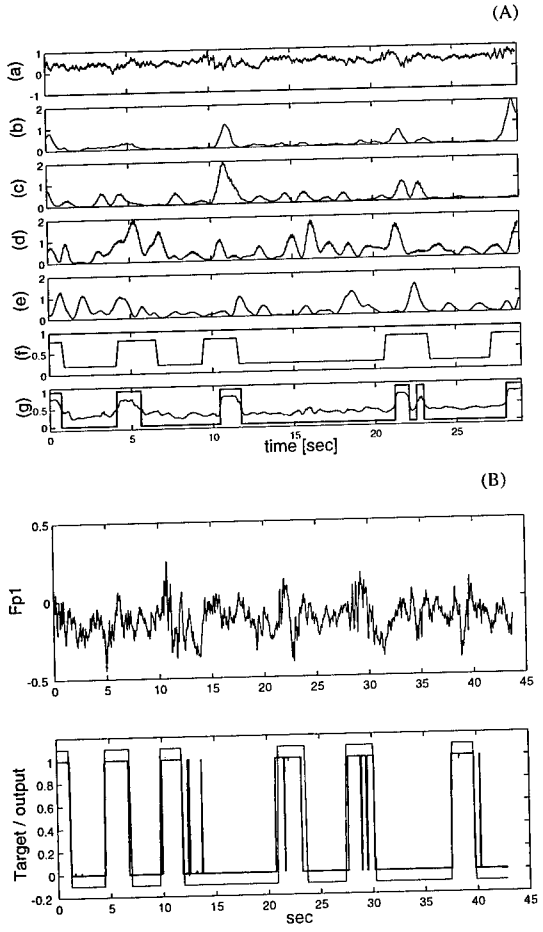


Figure 3: Segmenation of discontinuous EEG (electrode Fp_1, 128 Hz sampling rate) using an MLP combined with FIR based (A) and IIR based (B) preprocessing. The subplots (A):(b)–(e) show the outputs of the adapted FSQFs constructed with FIR bandpass filters. The lower subplots of both figures show the target time series and the output of the neural classifier (bold). (In (B) the target has been scaled for improved visibility.)

selective ASPUs has been proposed. Whereas FIR–based systems offer a distortionless transmission, the adaptation of their frequency parameters is usually more complicated than for IIR systems. Furthermore, the latter provide more efficient solutions for the construction of extremely narrow passbands.

For a signal classification task, the efficiency of the new approach has been demonstrated. Systems of similar structure seem to be applicable for a number of problems where either events of similar, though interindividually different characteristics or with (slowly) time–varying features have to be detected. In both cases a moderate learning effort could save a significant amount of manual evaluation.

References

[1] M. Arnold, A. Doering, H. Witte, J. Doerschel, and M. Eiselt. Use of adaptive Hilbert transformation for EEG segmentation and calculation of instantaneous respiration rate in neonates. *J. Clin. Monitoring*, 12:43–60, 1996.

[2] A. D. Back and A. C. Tsoi. FIR and IIR synapses, a new neural network architecture for time series modeling. *Neural Computation*, 3(3):375–385, 1991.

[3] A.G. Deczky. Synthesis of recursive digital filters using the minimum P error criterion. *IEEE Trans. Audio Electroacoustics*, 20:257–263, 1972.

[4] A. Doering and H. Witte. Extended neural networks for signal detection and classification: An approach for simultaneous optimization of parameterized preprocessing and neural networks. In J. Brender et. al., editor, *Medical Informatics Europe 96*, pages 977–981. IOS Press, 1996.

[5] B.H. Jansen and P.R. Desai. K-complex detection using multi-layer perceptrons and recurrent networks. *International Journal of Bio–Medical Computing*, 37(3):249–257, 1994.

[6] T.W. Parks and J.U. McClelland. Chebycheff approximation for nonrecursive digital filters with linear phase. *IEEE Trans. ET*, 19:189–194, 1972.

[7] C.S. Pattichis and C.N. Schizas. Neural network models in EMG diagnosis. *IEEE Transactions on Biomedical Engineering*, 42(5):486–496, 1995.

[8] DeLiang Wang, Xiaomei Liu, and Stanley C. Anhalt. On temporal generalization of simple recurrent networks. *Neural Networks*, 9(7):1099–1188, 1996.

[9] H. Witte, J. Doerschel, and G. Griessbach. The combination of on–line preprocessing with neural network classification for EEG monitoring in neonates. *Beitr. Anaesth. Intens. Notfallmedizin*, 43(359–373), 1994.

WAVE PROPAGATION AS A NEURAL COUPLING MECHANISM: HARDWARE FOR SELF-ORGANIZING FEATURE MAPS AND THE REPRESENTATION OF TEMPORAL SEQUENCES

D. Ruwisch, B. Dobrzewski, and M. Bode
Universität Münster, Institut für Angewandte Physik,
Corrensstr. 2–4, D-48149 Münster, Germany.
E-mail: ruwisch@uni-muenster.de

Wave propagation within a "cortex" of neurons is introduced as a neural coupling mechanism. Using this effect for the control of the neural learning process, the network generates self-organizing feature maps. Additionally, wave propagation is used to influence the neural competition in representing the input of the network. By this means the network is able to represent temporal aspects of the input. Since apart from a global bus such a network only requires local interactions, connectivity is very low and a parallel hardware implementation suggests itself. Operation of a demonstration setup consisting of 16 neurons in digital technology is demonstrated by the representation of phoneme sequences.

1 INTRODUCTION

Wave propagation caused by reactive and diffusive processes is ubiquitous in nature. Such waves can be observed in several biological systems, e.g., among cells of the liver, muscles, in ova (for a review see [1]) or within networks of glial cells in the brain [2]. The latter example is especially interesting, since up to now it is still unclear wether and, if so, how glial cells participate in the information processing of the brain [3]. Also in the inanimate nature there are several examples of active media showing wave propagation, e.g., chemical and gas-discharge systems or semiconductor substrates (see e.g. [4]). In these cases diffusion is the spatial interaction that enables the propagation process.

Using an electrical network as active medium we have demonstrated that wave propagation can serve as neural coupling mechanism in self-organizing topographic feature maps [5]. This facilitates an efficient hardware realization of self-organizing neural networks. Apart from a data bus this hardware concept only needs local interactions. Thus, connectivity within the network is very low. Since intrinsic dynamic effects of active media are used for

0-7803-4256-9/97/$10.00

information processing, this concept may be called "synergetic" hardware implementation [6, 7]. Such a neural hardware is very interesting in view of technical applications, since self-organizing feature maps are used in various technical fields [8].

Furthermore, feature map generation is interesting from the biological point of view. Topographic feature maps can be observed in various parts of the mammalian cortex. There are, e.g., retinotopic, tonotopic or somatotopic maps (see e.g. [9]).

2 KOHONEN'S ALGORITHM

We focus on Kohonen's algorithm, which is the most perspicuous model of a self-organizing feature map. Each neuron, $N_{\mathbf{r}}$, of the network is located in a "cortex" at a position $\mathbf{r}$. Every neuron corresponds to a reference vector, $\mathbf{W_r}$, in an arbitrary feature space. When the neurons are presented with a feature vector, $\mathbf{U}$, a winner neuron N_{win} has to be determined. The winner neuron, which is located at cortical position $\mathbf{r}_{win}$, is thought to best represent the feature vector, $\mathbf{U}$:

$$\mathbf{r}_{win} = \arg\min(\|\mathbf{U} - \mathbf{W_r}\|). \tag{1}$$

Such a minimum detection is used not only in Kohonen's algorithm but also in several other neural network models. It can be performed in parallel by means of a suitable active medium [5]: Every neuron feeds a stimulus to the medium. This stimulus is a monotonously decreasing function of the feature space distance $\|\mathbf{U} - \mathbf{W_r}\|$, so that the highest stimulus is produced by the winner neuron designate. On increasing a global medium parameter, the highest stimulus causes an "ignition" of the active medium. Detection of this ignition terminates the parameter increase and determines the winner neuron.

The essential of Kohonen's algorithm is the learning step, which affects the winner neuron and its cortical neighborhood. Learning is an adaption, $\Delta\mathbf{W_r}$, of reference vectors, $\mathbf{W_r}$, in direction of the feature vector, $\mathbf{U}$. With increasing cortical distance,

$$d_{win}(\mathbf{r}) = \|\mathbf{r}_{win} - \mathbf{r}\|, \tag{2}$$

between a neuron at position $\mathbf{r}$ and the winner neuron at position $\mathbf{r}_{win}$, the relative amount of reference vector adaptation decreases. This interaction is described by the neighborhood function, $\eta(d_{win}(\mathbf{r}))$:

$$\Delta\mathbf{W_r} = \eta(d_{win}(\mathbf{r}))(\mathbf{U} - \mathbf{W_r}). \tag{3}$$

Feature vectors, $\mathbf{U}$, are thought to be stochastically presented while both height and width of the neighborhood function, $\eta(d_{\text{win}}(\mathbf{r}))$, are successively diminished.

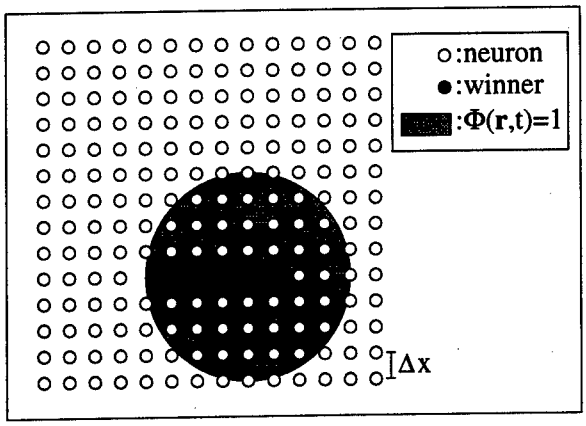


Figure 1: A wave front, $\Phi(\mathbf{r},t)$, starting at the position of the winner neuron, $\mathbf{r}_{win}$, at time t=0. It propagates within the cortex with veclocity c into the neighborhood of the winner neuron. Neurons that are reached by the wave front, i.e. that detect $\Phi(\mathbf{r},t)$=1, start executing the learning rule, Eq. (5).

3 WAVE PROPAGATION AS A COUPLING MECHANISM

Let us turn back to the synergetic implementation of Kohonen's network based on an active medium. If the medium shows suitable dynamic properties, a wave front propagates through the medium starting at time t=0 from the point of ignition, $\mathbf{r}_{win}$, i.e., the position of the winner neuron (Fig. 1). The simplest (although not very natural) form of such a wave front propagating with velocity c reads

$$\Phi(\mathbf{r},t) = \mathcal{H}(ct - \|\mathbf{r}_{win} - \mathbf{r}\|), \tag{4}$$

with the Heaviside function, $\mathcal{H}(x)$=0 if x<0, and $\mathcal{H}(x)$=1, otherwise.
This wave front triggers the learning process of each neuron it reaches:

$$\begin{aligned} \frac{d}{dt}\mathbf{W}_{\mathbf{r}}(t) &= 0 \qquad\qquad \text{if } \Phi(\mathbf{r},t) \leq 0.5, \\ \frac{d}{dt}\mathbf{W}_{\mathbf{r}}(t) &= \frac{1}{\tau}(\mathbf{U} - \mathbf{W}_{\mathbf{r}}(t)) \text{ if } \Phi(\mathbf{r},t) > 0.5, \end{aligned} \tag{5}$$

with a time constant, τ. This "local" learning rule can easily be realized in a synergetic manner by means of an RC-device consisting of a capacitor, C, and a resistor, R, yielding τ=RC. After a certain time, t_1, the medium is reset, i.e. $\Phi(\mathbf{r},t) \equiv 0$ if $t \geq t_1$. Within time t_1 the wave front has travelled a distance d_{max}=ct_1. Equation (5) yields a neighborhood function, η, in the Kohonen learing rule, Eq. (3), reading

$$\eta(d_{win}(\mathbf{r})) = 0 \qquad\qquad \text{if } d_{win}(\mathbf{r}) \geq d_{max}$$

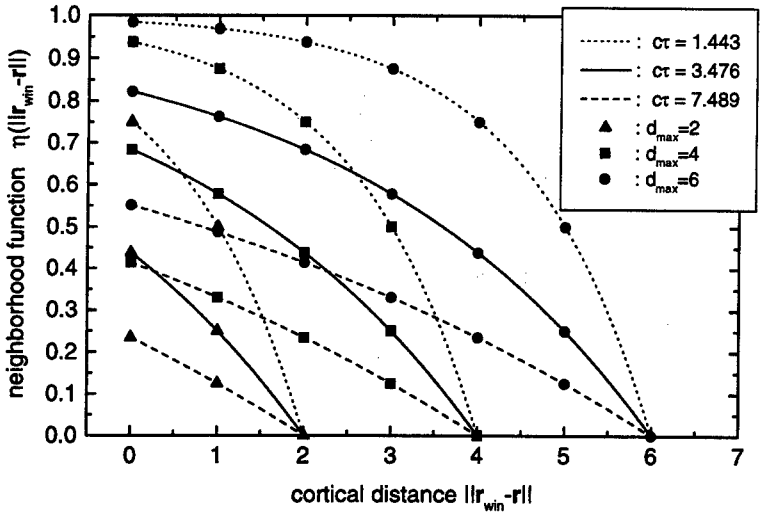


Figure 2: The neighborhood function, $\eta(\|\mathbf{r}_{win} - \mathbf{r}\|)$ (Eq. 6), for different values of the parameters d_{max} and $c\tau$. The values of $c\tau$ are equivalent to l=2, l=4, l=8 in the digital version (Eq. 13).

$$\eta(d_{win}(\mathbf{r})) = 1 - \exp\left(\frac{d_{win}(\mathbf{r}) - d_{max}}{c\tau}\right) \text{ if } d_{win}(\mathbf{r}) < d_{max} \qquad (6)$$

The shape of the neighborhood function, η, is shown in Fig. 2 for different values of the parameters d_{max} and $c\tau$.

4 REPRESENTATION OF TEMPORAL SEQUENCES

Different attempts have been made to extend or modify the self-organizing feature map in order to represent temporal aspects of the presented features. This is a very important task if the network is supposed to process context information, e.g. in speech processing systems [10]. In general, such a task demands a representation of the near past. This can be achieved by means of a time-delay architecture, which enables the system to process a certain amount of former inputs (for a review see e.g. [11]). One can apply this technique to the input of a standard Kohonen feature map: A number of time-delayed input vectors are concatenated to a larger vector that serves as feature vector, $\mathbf{U}$. This method was succesfully used to improve the recognition of transient phonemes in a speech recognition system [10].

Another idea to represent temporal dynamics is to use an independent, fully interconnected network, which learns the transitions within the cortex of the feature map ([12]). Such a network is even capable of reproducing a trained input sequence. A more implicit representation of the past is realized by

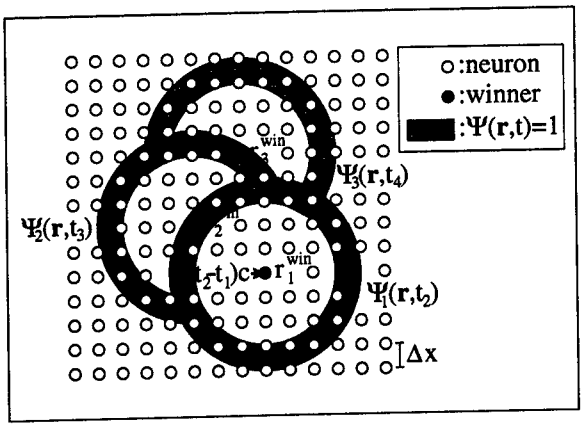


Figure 3: Three waves $\Psi_i(\mathbf{r}, t)$ in the cortex according to Eq. (8) started by three subsequent winner neurons at cortical positions $\mathbf{r}_i^{win}$. Subsequent winner neurons are forced by the wave to be situated in a certain cortical neighborhood of the respective preceding winner neuron.

providing the neurons of the feature map with a retention and decay of their activity. If such information on former activity, i.e., being a winner neuron, influences the present determination of the winner neuron, then the network becomes able to represent temporal sequences [13].

Following an idea of Euliano and Principe [14], wave propagation can be utilized to process information on the past so that temporal coherence is represented in the feature map. Propagation and interference of waves, that are attenuated over space and time, influence the neural competition: Determination of the winner neuron is modified with the result, that temporal neighborhood of feature vectors in a sequence may lead to an adjacent representation of these vectors in the feature map. Euliano and Principe choose a predetermined direction of wave propagation. We will restrict propagation in a less rigid and more natural way leading to some advantages. Additionally, we omit attenuation and interference of waves for the sake of an easier hardware implementation.

Consider a wave, $\Psi_i(\mathbf{r}, t)$, with a concentric wave crest of certain width, b, propagating through the cortex. It starts at time t=t_i at the position of the winner neuron, $\mathbf{r}_i^{win}$. The index, i, numbers the sequence position of the corresponding feature vector, $\mathbf{U}_i$, starting with i=1. Wave propagation ends at time t_{i+1}, when the next feature vector, $\mathbf{U}_{i+1}$, is presented. Using the abreviations Eq. (2) and

$$t_i' = t - t_i \tag{7}$$

such a wave reads

$$\Psi_i(\mathbf{r},t) = \mathcal{H}(ct_i' - d_{win}(\mathbf{r}))\mathcal{H}(d_{win}(\mathbf{r}) - ct_i' + b)H(\mathbf{r},t) \tag{8}$$

where $\mathcal{H}$, again, is the Heaviside function. The "History function", $H(\mathbf{r},t)$, restricts wave propagation with respect to the past. It is initialized at the beginning of a new sequence with

$$H(\mathbf{r},t_1) = 1, \text{ for all } \mathbf{r}. \tag{9}$$

Afterwards, it is set to zero at a position, $\mathbf{r}$, if the crest of a wave has left that position, i.e.,

$$H(\mathbf{r},t) \to 0 \text{ if } \Psi_i(\mathbf{r},t) \to 0 \tag{10}$$

and remains zero for the rest of the sequence.

As a result, a subsequent wave does not propagate into a region that has been passed by an earlier wave during the same sequence. This resembles the refractory phase of real neural tissue. Even more natural dynamics would allow for a decay of $H(\mathbf{r},t)$ over time. Moreover, this could be understood as an explicit measure of time within the network (compare [13]).

Wave propagation according to Eq. (8) and Eq.(10) is sketched in Fig. 3. The final states of three subsequent waves, i.e., $\Psi_i(\mathbf{r},t_{i+1})$, i=1..3, are shown. In the final wave region the probability for a neuron to win the competition and to become the winner neuron is enhanced. Equation (1) of the Kohonen algorithm is changed to

$$\mathbf{r}_i^{win} = \arg\min(\|\mathbf{U_i} - \mathbf{W_r}\| - \beta\,\Psi_{i-1}(\mathbf{r},t_i)). \tag{11}$$

Only the first winner (i=1) is conventionally determined, i.e., $\Psi_0 \equiv 0$ in Eq. (11). In this case Eq. (11) equals the original Kohonen Eq. (1). On the other hand, if β is large, the position, $\mathbf{r}_{i+1}^{win}$, of the next winner neuron is forced to be in the region $\Psi_i(\mathbf{r},t_{i+1})$=1 (Fig. 3), i.e., in a certain neighborhood of the preceding winner neuron. The later the succeeding feature vector is presented, the farther away from the predecessor it will be represented. Since this is a competitive ordering principle, there may arise contradictions to a purely topographic mapping (β=0), if very different feature vectors are neighbors in the sequence or if very similar feature vectors appear far from each other in the sequence. However, by tuning β one can choose what more attention is paid to: feature similarity or temporal coherence.

Figure 3 shows a parameter set, where the distance, $(t_{i+1}-t_i)c$, being covered by the wave is distinctly larger than the cortical distance of neighboring neurons, Δx. In this situation it is possible to represent a noisy feature cluster with more than one neuron, i.e., with improved resolution. As a consequence, the subsequent feature vector will be represented relatively far away from the predecessor. Within the regions of the single, noisy feature cluster, the mapping is purely topographic. A disadvantage in such a case is

that the total length of a sequence that can be entirely represented is smaller than in the case $(t_{i+1} - t_i)c \approx \Delta x$.

If a sequence is too long to be entirely represented, i.e., the wave has completely left the cortex ($\Psi_i(\mathbf{r}, t)$=0 over the whole cortex), Eq. (11), again, becomes identical with Eq. (1). In this case, the next winner is determined independently from its predecessor (standard Kohonen). Thus, if the new winner neuron is not situated in a refractory region, the sequence is automatically devided into two independent subsequences. Otherwise, no wave is started and the system determines another winner in the standard Kohonen manner.

In the described concept there are two independent processes being governed by wave propagation: 1) Influencing the winner election in order to represent temporal coherence (Eq. (11)); 2) Control of the learning process of the neurons (Eq. (5)). Both processes can be governed by the same wave since learning can also be triggered by a wave according to Eq. (8) instead of Eq. (4). This can be done in a way that learning is not affected by the history function, Eq. (10); i.e., refractory neurons can learn as usual. The only restriction is that the parameter d_{max} in the neighborhood function, Eq. (6), be limited to

$$d_{max} \leq (t_{i+1} - t_i)c \tag{12}$$

5 DIGITAL HARDWARE IMPLEMENTATION

The described concept was developed in view of a digital hardware realization. In such an implementation Eq. 5 is replaced by its time discrete analogue,

$$\Delta \mathbf{W_r} = \frac{\mathbf{U} - \mathbf{W_r}}{l}, \tag{13}$$

with a parameter l. The wave is replaced by a spatially discrete "domino-effect". Eq. 13 is executed by the concerned neurons after each propagation step of the domino-effect. Owing to this space- and time-discrete procedure distance measurement in the cortical space is changed from Euclidian to Manhattan distance. The distance of adjacent neurons is set to $\Delta x \equiv 1$. The neighborhood function, $\eta(d_{win}(\mathbf{r}))$, is the same as given in Eq. 6 with the transformation

$$c\tau = \left(\ln\left(\frac{l}{l-1}\right)\right)^{-1} \tag{14}$$

Restricting l to a power of two allows to carry out division by l (Eq. 13) in a very simple way. Since the whole concept does not require any multiplication and because connectivity within the network is very low, the concept is well-suited for chip integration. As learning is done stepwise (Eq. 13)) rather than continuously (Eq. 5) the possible range of d_{max} is extended over the limit given in Eq. (12) to

$$d_{max} \leq (t_{i+1} - t_i)c + \Delta x. \tag{15}$$

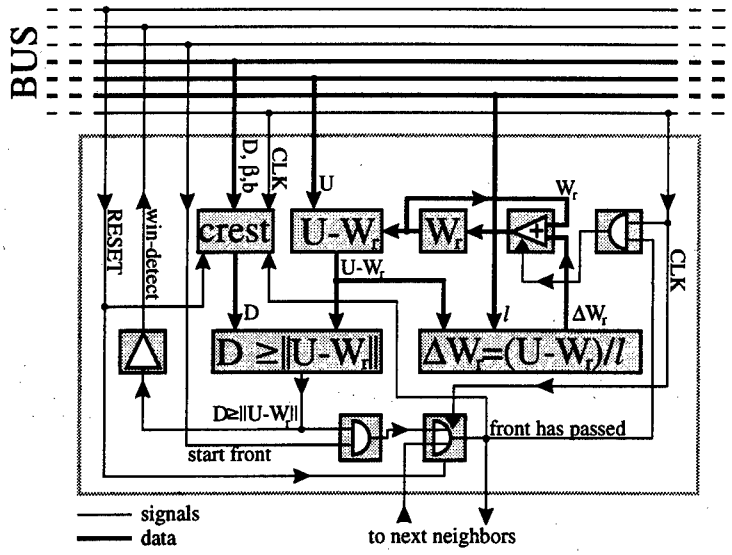


Figure 4: A single neuron N_r of the digital setup. A feature vector, $\mathbf{U}$, is presented to all neurons via a global bus. The winner neuron is determined by means of a binary search with a sequence of numbers D: if there is a neuron detecting $D \geq \|\mathbf{U} - \mathbf{W}_i\|$, D is diminished, otherwise it is increased by the current power of two. When the search is over, a winner neuron is found and a new front is started by this neuron. If the the neuron was passed by the former front not longer than b clock cycles before, it belongs to the crest region and increases the numbers D by β. Thus, a winner election corresponding to Eq. (11) results. After a winner neuron is found, each neuron detecting that the front has passed executes the learning rule (13) with every clock cycle. The front is realized by clocked OR-units being interconnected with the next cortex-neighbors: The output is connected to the inputs of the OR-units of the neighboring neurons and vice versa. After n clock cycles the OR-units are reset and a new front is started by the presentation of the subsequent feature vector.

Operation is demonstrated with phoneme sequences processed by a demonstration setup which consists of 16 neurons in a linear chain. Schematic operation of a single neuron is sketched in Fig. 4. Each neuron is emulated by a PIC16C84 microcontroller. The components of the feature ($\mathbf{U}$) and reference vectors ($\mathbf{W_r}$) are stored as 8bit values. The acoustic signal is sampled (rate: 14kHz) and Fourier-transformed (512 points in 36.7 ms) by a digital signal processor TMS320C26. The resulting spectrum is grouped into three frequency channels (0.6–1.0 kHz, 1.0–3.5 kHz, 3.5–7.4 kHz). The energy of each single channel contributes one component to a 3-dimensional feature vector, $\mathbf{U}$, which, additionally, is normalized. Each feature vector corresponds to one spoken phoneme, independently of its natural length.

In the 1-dim setup, the wave travels in both possible directions after it has

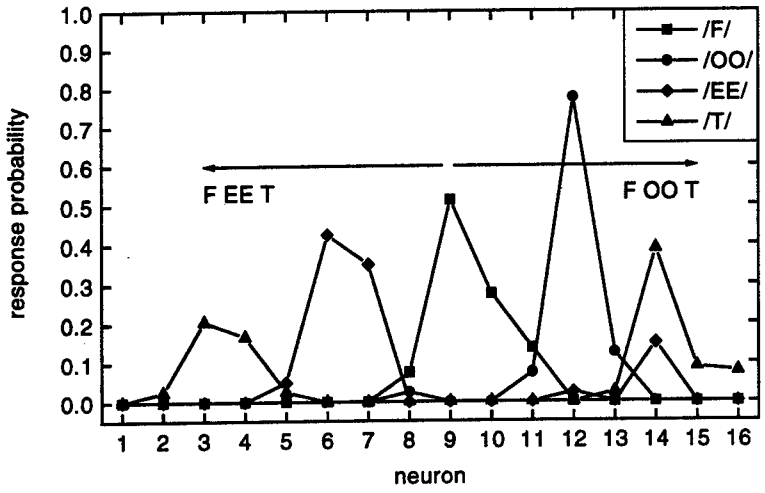


Figure 5: Representation of the two phoneme sequences "F OO T" and "F EE T" by the 16 neurons of the digital demonstration setup. The parameters of the experiment are β=765 (3×8bit, i.e. β is "large"), ct_i'=3 and b=2 (Eq. 8, compare Fig. 3) and l=16 (Eq. 13). The 16 reference vectors (3-dim), $\mathbf{W}_r$, are initialized with random numbers and learning is performed as follows: Ten presentations of each sequence (random order) with parameter d_{max}=4, 20 presentations with d_{max}=3, and 30 with d_{max}=1. (Eq. 6, compare Fig. 2). The probability distributions (conditional probabilities p(neuron|phoneme)) are obtained with 40 presentations of each sequence

been started at the position, $\mathbf{r}_1^{win}$, by the first winner neuron of a sequence. Owing to the refractory behaviour of the neurons, the second feature vector of a sequence selects a direction. The initial variability allows to represent two (in a 2-dim network even more) sequences with the same beginning. This is shown in Fig. 5 for the two 3-phoneme sequences "F OO T" and "F EE T" after 60 random presentations of each sequence. As a result of the self-organizing process the neurons are arranged in clusters. Each cluster represents a single noisy phoneme, as expected in the Kohonen model. Additionally, adjacent clusters represent subsequent phonemes in order to represent the temporal order in each sequence. The initial "F" of both sequences is represented in common in the middle of the network, whereas the final "T"s of each sequence are represented twice by each end of the network.

References

[1] T. Meyer, "Cell signaling by second messenger waves" **Cell**, vol. 64, pp. 675–678 (1991).

[2] A. H. Cornell-Bell, S. M. Finkenbeiner, M. S. Cooper, S. J. Smith, "Glutamate induces calcium waves in cultured astrocytes: Long-range glial signaling," **Science**, vol. 247, pp. 470–473 (1990).

[3] B. A. Barres, "New roles for glia," **J. Neurosci.**, vol. 11, pp. 3685–3694 (1991).

[4] F.-J. Niedernostheide, M. Arps, R. Dohmen, H. Willebrand, H.-G. Purwins, "Spatial and Spatio-Temporal Patterns in pnpn Semiconductor Devices," **Phys. Stat. Sol. B**, vol. 172, pp. 249–266 (1992).

[5] D. Ruwisch, M. Bode, H.-G. Purwins, "Parallel Hardware Implementation of Kohonen's Algorithm With an Active Medium," **Neural Networks**, vol. 6, pp. 1147–1157 (1993).

[6] H. Haken, "Principles of brain functioning," **Springer Series in Synergetics**, vol. 67 (Springer, Berlin, 1996).

[7] D. Ruwisch, M. Bode, H.-J. Schulze, F.-J. Niedernostheide, "Synergetic hardware concepts for self-organizing neural networks," **Lecture Notes in Physics**, vol. 476, pp. 194–212. Eds.: J. Parisi, S. C. Müller, W. Zimmermann (Springer, Berlin, 1996).

[8] T. Kohonen, **Self-Organizing Maps** (Springer, Berlin, 1995).

[9] G. D. Schott, "Penfield's homunculus: a note on cerebral cartography," **J. Neurology, Neurosurgery, Psychiatry**, vol. 56, pp. 329–333 (1993).

[10] J. Kangas, "Time-Delayed Self-Organizing Maps," **Proceedings of the international joint conference on neural networks (IJNN)**, vol. 2, pp. 331–336 (1990).

[11] M. C. Mozer, "Neural net architectures for temporal sequence processing," **Predicting the future and understanding the past.** A. Weigend and N. Gershenfeld (Eds.), (Addison Wesley, 1993).

[12] A. Hoekstra, M. F. J. Drossaers, "An extended Kohonen Feature Map for Sentece Recognition," *Proceedings of the international conference on neural networks (ICANN93)*, pp. 404–407 (1993).

[13] D. L. James, R. Miikulainen, "SARDNET: A self-organizing feature map for sequences," **Advances in neural processing systems**, Eds.: G. Tesauro, D. S. Touretzky, T. K. Leen, vol. 7 (1995).

[14] N. R. Euliano, J. C. Principe, "Spatio-Temporal Self-Organizing Feature Maps," **Proceedings of the international conference on neural networks (ICNN)** vol. 4, pp. 1900–1905 (1996).

On-line Adaptive Algorithms in Non-stationary Environments Using a Modified Conjugate Gradient Approach

Andrzej CICHOCKI Bruno ORSIER Andrew BACK
Shun-ichi AMARI
RIKEN Frontier Research Program,
Brain Information Processing Group,
Hirosawa 2-1, Wako-shi, Saitama 351-01, JAPAN

Phone: +81-48-465-2645, Fax: +81-48-462-4633, E-mail:
cia@zoo.riken.go.jp

Abstract

In this paper we propose novel computationally efficient schemas for a large class of on-line adaptive algorithms with variable self-adaptive learning rates. The learning rate is adjusted automatically providing relatively fast convergence at early stages of adaptation while ensuring small final misadjustment for cases of stationary environments. For non-stationary environments, the algorithms proposed have good tracking ability and quick adaptation to new conditions. Their validity and efficiency are illustrated for a non-stationary blind separation problem.

Keywords: Adaptive on-line learning algorithms, Blind equalization, Blind separation of sources from instantaneous mixture.

1 INTRODUCTION

The problem of optimal updating of a learning rate (step size) is a key problem encountered in a wide class of learning algorithms. Many of the research work related to this problem is devoted to batch and/or supervised algorithms. Many various techniques like conjugate gradient, quasi-Newton methods and Kalman filters have been applied. However, relative little work has been devoted to this problem for on-line adaptive algorithms [1, 8, 12, 13].

A large class of on-line learning algorithms (both supervised and unsupervised) used for training neural networks or nonlinear adaptive systems can

0-7803-4256-9/97/$10.00

be expressed in general form as [2, 8, 12, 13]:

$$\boldsymbol{\theta}(k+1) = \boldsymbol{\theta}(k) - \eta(k)\tilde{\mathbf{g}}(k) \quad (k = 0, 1, 2, ...), \tag{1}$$

where k is the iteration index, $\boldsymbol{\theta}(k) = [\theta_1(k), \theta_2(k), ..., \theta_n(k)]^T$ is the n-dimensional vector of updated unknown parameters, $\eta(k) > 0$ is a learning rate (step size), and $\tilde{\mathbf{g}}(k) = \tilde{\mathbf{g}}(\boldsymbol{\theta}(k), \mathbf{x}(k), \mathbf{y}(k)) = [\tilde{g}_1(k), \tilde{g}_2(k), \ldots, \tilde{g}_n(k)]^T$ is a nonlinear function depending on $\boldsymbol{\theta}(k)$ and $\mathbf{x}(k), \mathbf{y}(k)$ (input and output signals respectively).

In system identification problems for example, the algorithm defined by Eq. (1) is termed "supervised" due to the availability of an instantaneous output error or desired output signal. In such cases, $\tilde{\mathbf{g}}(k)$ is a function of the error and can be considered as the instantaneous gradient (which is a rough approximation of true gradient) of a loss (also called cost, error or energy) function $\mathcal{J}(\boldsymbol{\theta})$, i.e.:

$$\tilde{\mathbf{g}}(k) = \nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}, k) = \left[\frac{\partial \mathcal{J}(k)}{\partial \theta_1}, \frac{\partial \mathcal{J}(k)}{\partial \theta_2}, ..., \frac{\partial \mathcal{J}(k)}{\partial \theta_n} \right]^T. \tag{2}$$

However in many applications like blind separation of sources, the true loss function $\mathcal{J}(\boldsymbol{\theta})$ is not available or its evaluation is too time consuming. Moreover, if $\tilde{\mathbf{g}}(k)$ is not a function of the error, the algorithm is termed "unsupervised".

It is well known that the final misadjustment (often defined in terms of the mean square error MSE) increases as the learning rate η increases. However, the convergence time increases as the learning rate decreases [1]. For this reason it is often assumed that the learning rate η is a very small positive constant, either fixed or exponentially decreasing to zero as time goes to infinity. Such an approach leads to relatively slow convergence speed and/or low performance and is not suitable for non–stationary environments.

This inherent limitation of on-line adaptive algorithms represented by (1) imposes a compromise between two opposing fundamental requirements of small misadjustment and fast convergence demanded in most applications, especially in non-stationary environments [1, 12]. As a result, it is desirable to find an alternative method to improve the performance of such algorithms. Most of the works have been devoted to variable step size LMS-type (supervised delta rule) algorithms [11, 15]. In this paper we will consider a more general case which includes unsupervised and/or supervised on-line algorithms.

2 SELF-ADAPTIVE VARIABLE STEP SIZE ALGORITHMS

Amari in his fundamental work [1] analyzed the dynamic behavior of $\boldsymbol{\theta}(k)$ in the neighborhood of the optimal $\boldsymbol{\theta}_*$ and obtained analytical formulas for the convergence speed to $\boldsymbol{\theta}_*$ and the fluctuation of $\boldsymbol{\theta}(k)$ around $\boldsymbol{\theta}_*$. Moreover, he

proposed an efficient method for a variable learning rate [1]. In this paper we extend Amari's idea of learning of learning rate by proposing robust, variable step-size on-line algorithms. The main objective of this paper is to develop a simple and efficient algorithm which enables automatic updating of the learning rate. This is especially suited to non-stationary environments.

Recently, on the basis of Amari's works we have developed a new family of on-line algorithms with an adaptive learning rate suitable for non-stationary environments [8, 9]:

$$\begin{aligned}
\boldsymbol{\theta}(k+1) &= \boldsymbol{\theta}(k) - \eta(k)\hat{\mathbf{g}}(k), && (3)\\
\hat{\mathbf{g}}(k) &= (1-\rho_2)\hat{\mathbf{g}}(k-1) + \rho_2\tilde{\mathbf{g}}(k), && (4)\\
\eta(k) &= (1-\rho_1)\eta(k-1) + \rho_1\beta\psi(||\hat{\mathbf{g}}(k)||) && (5)\\
&\text{or} \quad [1-\rho_1\eta(k-1)]\eta(k-1) + \rho_1\beta\eta(k-1)\psi(||\hat{\mathbf{g}}(k)||), && (6)
\end{aligned}$$

where $0 < \rho_1 < 1$, $0 < \rho_2 < 1$, $\beta > 0$ are fixed coefficients and $\psi(||\hat{\mathbf{g}}(k)||)$ is a nonlinear function defined, e.g., as $\psi(||\hat{\mathbf{g}}(k)||) = \frac{1}{n}\sum_{i=1}^{n}|\hat{g}_i(k)|$ or $\psi(||\hat{\mathbf{g}}(k)||) = \tanh(\frac{1}{n}\sum_{i=1}^{n}\hat{g}_i(k)^2)$, with $\hat{g}_i(0) = \tilde{g}_i(0)$. It should be noted that Eq. (6) X1has been obtained from Eq. (5) by simply replacing the fixed ρ_1 by a self-adaptive term $\rho_1\eta(k-1)$. The above algorithms are related (especially Eq. (6)) to the very recent research works of Murata et al and Sompolinsky et al [12, 13]). It is interesting to note that Eqs.(4)–(5) describe simple first-order low-pass filters (LPFs) with cut-off frequencies determined by the parameters ρ_1 and ρ_2. The even nonlinear function ψ is introduced to limit the maximum value of the gradient norm $||\hat{\mathbf{g}}(k)||$ and maximal value of the gain β is constrained to ensure stability of the algorithm [8, 9].

It should be noted that for fixed η the system behaves in such a way that parameters $\theta_i(k)$ never achieve steady state but will fluctuate around some equilibrium point. In order to reduce such fluctuations we have employed low-pass filters. Intuitively, the self-adaptive system described by Eqs.(2)-(5) operates as follows. If gradient components $\tilde{g}_i(k)$ have local average (mean) values which differ from zero (which are extracted by the first LPF's) then the learning rate $\eta(k)$ is decreasing or increasing to a certain value determined by gain β and norm of gradient components. However, during the learning process $|\hat{g}_i(k)|$ decreases to zero, i.e. after some time $\tilde{g}_i(k)$ starts to fluctuate around zero, then $\eta(k)$ decreases also to zero exponentially as desired. If some rapid changes occur in the system then $|\hat{g}_i(k)|$ suddenly increases and consequently $\eta(k)$ also rapidly increases from a small value so that the system is able automatically to adapt quickly to new environments.

The learning algorithm (3)-(6) employs global learning rate, i.e. the same variable step size $\eta(k)$ for all the weights θ_i. In order to improve performance we can use local learning rates, i.e. each parameter θ_i can have an individual learning rate $\eta_i(k)$ as follows [8]:

$$\begin{aligned}
\theta_i(k+1) &= \theta_i(k) - \eta_i(k)\tilde{g}_i(k),\\
\eta_i(k) &= (1-\rho_1)\eta_i(k-1) + \rho_1\beta\psi(|\hat{g}_i(k)|)\\
&\text{or} \quad \eta_i(k-1) + \rho_1\eta_i(k-1)\left[\beta\psi(|\hat{g}_i(k)|) - \eta_i(k-1)\right],
\end{aligned}$$

$$\hat{g}_i(k) = (1-\rho_2)\hat{g}_i(k-1) + \rho_2 \tilde{g}_i(k).$$

We have found that these algorithms work with high efficiency and they are easily implemented in VLSI technology. However, they require the proper selection of three parameters (ρ_1, ρ_2, β). Although the above algorithms are robust with respect to the values of these parameters, their optimal choice is problem dependent. Moreover, the algorithm can still be too slow for some applications. In order to increase the convergence speed and improve overall performance we propose in this paper a new learning strategy which is based on the modified conjugate gradient (CG) approach. We would like to emphasis here, that although CG techniques are very well known and have been applied successfully to many optimization problems, most of these applications are related to batch and supervised learning algorithms. To our knowledge the CG approach has not been deeply investigated for on-line adaptive algorithms, especially, for unsupervised algorithms when cost (loss) functions are not explicitly available. In other words, the problem is formulated as follows: on basis of available on-line learning rule it is necessary to develop efficient algorithms (self-adaptive systems) which provide self-adaptive adjustment of learning rate and ensure high convergence speed and small misadjustment error.

In related works other modified conjugate gradient algorithms have been proposed for adaptive filtering (see e.g. [6]). However, these proposals are limited to supervised LMS type adaptive algorithms. In this paper we propose a more general approach which can applied to any kind of on-line algorithms in the form of Eq.(1) without the knowledge of the Hessian or line search.

3 A MODIFIED CONJUGATE GRADIENT ADAPTIVE ALGORITHM

In this section we propose a new scheme which extends our previous algorithms (2)-(6) [8]. Consider first the standard CG algorithm [10, 6] shown below, and which is suitable only for batch learning problems since the cost function $\mathcal{J}$ and its exact gradient $\mathbf{g}$ must be available:

$$\boldsymbol{\theta}(k+1) = \boldsymbol{\theta}(k) + \eta(k)\mathbf{d}(k), \tag{7}$$

with
$$\mathbf{d}(k) = -\mathbf{g}(k) + \alpha(k)\mathbf{d}(k-1), \tag{8}$$

$$\eta(k) = \arg\min_{\eta} \mathcal{J}(\boldsymbol{\theta}(k) + \eta\mathbf{d}(k)) \quad \text{(LS procedure)}, \tag{9}$$

and
$$\alpha(k) = \frac{(\mathbf{g}(k) - \mathbf{g}(k-1))^T \mathbf{g}(k)}{\mathbf{g}(k-1)^T \mathbf{g}(k-1)} \quad \text{(PR formula)}, \tag{10}$$

$$\text{or} \quad \frac{\mathbf{g}(k)^T \mathbf{g}(k)}{\mathbf{g}(k-1)^T \mathbf{g}(k-1)} \quad \text{(FR formula)}, \tag{11}$$

where in this case k is an index in parameter space, $\mathbf{d}(k)$ is a search direction, LS means "line-search", PR means "Polak-Ribiere", FR means "Fletcher-Reeves". It can be shown that if the cost function is a n-order quadratic

function, this CG algorithm finds the minimum in n steps. In extensions of CG to non-quadratic functions, $\alpha(k)$ is reset to zero every R (typically $R \cong n$) iterations in order to improve the rate of convergence near the minimum [10]. Consequently in the case of such a "restart", Eq. (8) becomes:

$$\mathbf{d}(k+1) = -\mathbf{g}(k+1). \tag{12}$$

Unfortunately, the above algorithm cannot be applied directly to learning rule (1) since we assume that the loss function $\mathcal{J}(\boldsymbol{\theta})$ is not available. In this case the line search to find the optimal value of $\eta(k)$ in each iteration step can not be performed. Moreover, the true value of gradient $\mathbf{g}(k)$ is not directly available in on-line adaptive algorithms and must be estimated in each iteration step.

For this reason we propose a modified CG approach in which a time consuming line search procedure is replaced by low-pass filtering of the gradient norm. Here, we can use any norm (e.g. Euclidean norm) which converts a vector to a scalar (see Eqs.(3)-(6)). In other words, instead of line search we have applied a simple low-pass (averaging) technique to estimate the learning rate $\eta(k)$.

The parameters in the update recursion are given by (see Fig.1 (a)):

$$\begin{aligned}
\boldsymbol{\theta}(k+1) &= \boldsymbol{\theta}(k) + \eta(k)\mathbf{d}(k), && (13)\\
\mathbf{d}(k) &= -\hat{\mathbf{g}}(k) + \alpha(k)\mathbf{d}(k-1), && (14)\\
\eta(k) &= (1-\rho_1)\eta(k-1) + \beta\rho_1\psi(||\mathbf{d}(k)||), && (15)\\
&\text{or}\quad [1-\rho_1\eta(k-1)]\eta(k-1) + \rho_1\beta\eta(k-1)\psi(||\mathbf{d}(k)||), && (16)
\end{aligned}$$

where $0 < \rho_1 < 1$ is a small positive constant, $\alpha(k)$ is an adaptive coefficient computed on basis of the Polak-Ribiere or Fletcher-Reeves formula with resetting to zero for every R iterations or if $\alpha(k)$ achieves a value greater than a specified threshold (typically a value between 1 and 2). An estimate $\hat{\mathbf{g}}(k)$ of the true gradient $\mathbf{g}(k)$ is computed using one of the following formula:
1) Recurrent formula realized by a first-order IIR low-pass filter (see Fig.1 (b)):

$$\hat{\mathbf{g}}(k) = (1-\rho_2)\hat{\mathbf{g}}(k-1) + \rho_2\tilde{\mathbf{g}}(k) \tag{17}$$

2) Sliding window realized by M-th order FIR low-pass filter which in the special case performs simple averaging (possibly with forgetting factor γ) (see Fig.1 (c)):

$$\hat{\mathbf{g}}(k) = \frac{1}{M}\sum_{i=0}^{M-1}\gamma^i\tilde{\mathbf{g}}(k-i) \tag{18}$$

The on-line learning algorithm (13)-(18) has useful circuit and signal processing interpretations (see Fig.1 (a), (b) and (c)). Equations (14)-(17) could be considered as first order low-pass discrete-time filters. From these interpretations many possible extensions or generalizations follow. For instance, instead of first-order low-pass filters we could employ second or higher order IIR or

FIR filters. Moreover, coefficients ρ_1, ρ_2 and $(1-\alpha(k))$ can be interpreted as parameters corresponding to cut-off frequency of low pass filters. Instead of filter with fixed cut-off frequencies we could use adaptive filters with adjustable cut-off frequency, e.g. we could assume that $\rho_1(k) = \rho_1\eta(k-1)$ (see Eq. (6) [8, 12]).

Our new algorithm (13)-(18) offers the following advantages: (i) high convergence speed due to the optimization based on the conjugate gradient approach; (ii) low complexity due to the replacement of a costly line search procedure by a much simpler computation of the step size (Eq. (15)); (iii) adaptive learning rate suitable for non-stationary environments.

4 COMPUTER SIMULATION EXPERIMENTS

In order to confirm the validity and the performance of the developed on-line adaptive algorithms we have tested them on a number of specific problems, in particular blind equalization/deconvolution and blind separation problems [3, 4, 8]. Due to lack of space we give here only an illustrative example for blind separation of instantaneous mixture of sources.

The robust unsupervised on-line learning algorithm for the standard blind separation of m sources can be formulated in vector form [4, 8, 7, 5, 14]:

$$\mathbf{w}_i(k+1) = \mathbf{w}_i(k) - \eta_i(k)\tilde{\mathbf{g}}_i(k), \tag{19}$$

where $\tilde{\mathbf{g}}_i(k) = -\mathbf{w}_i(k) + \mathbf{f}[\mathbf{y}(k)]\mathbf{y}^T(k)\mathbf{w}_i(k)$, $\mathbf{w}_i = [w_{i1}, w_{i2}, ..., w_{im}]^T$, $(i = 1, 2, ..., m)$, $\mathbf{y}(k) = \mathbf{W}(k)\mathbf{x}(k)$, $\mathbf{x}(k) = \mathbf{A}\mathbf{s}(k)$, $f(y) = y^3$, $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_m]$. In this formulation there are m local learning rates η_i; a simplified formulation consists in using only one global learning rate η, so that we obtain an on-line algorithm of the form (1) as

$$\boldsymbol{\theta}(k+1) = \boldsymbol{\theta}(k) - \eta(k)\tilde{\mathbf{g}}(k), \tag{20}$$

where concatenated vectors are defined as $\boldsymbol{\theta}(k) = [\mathbf{w}_1^T(k), \mathbf{w}_2^T(k), \ldots, \mathbf{w}_m^T(k)]^T$ and $\tilde{\mathbf{g}}(k) = [\tilde{\mathbf{g}}_1^T(k), \tilde{\mathbf{g}}_2^T(k), \ldots, \tilde{\mathbf{g}}_m^T(k)]^T$. Thus our learning schema (3)-(6) and (13)-(15) can be directly applied to this problem. The key aspect of the problem we consider is that the elements of the mixing matrix $\mathbf{A}$ are not fixed but change rapidly or drift slowly over time.

Illustrative example: we consider the following synthetic source signals (assumed to be unknown to the algorithm) sampled with sampling period $\Delta t = 2 \cdot 10^{-4} s$:

$$\begin{aligned} s_1(k) &= \sin(200\pi k\Delta t)\cos(300\pi k\Delta t), \\ s_2(k) &= \operatorname{sign}(\cos(400\pi k\Delta t + 50\sin(30\pi k\Delta t))) \end{aligned} \qquad \mathbf{A}_1 = \begin{bmatrix} 0.3 & 0.5 \\ -0.8 & 0.1 \end{bmatrix}$$

First experiment (see fig. 2 (a)): Sensors signals are obtained by mixing the source signals with examplar mixing matrix $\mathbf{A}_1$ during the first second

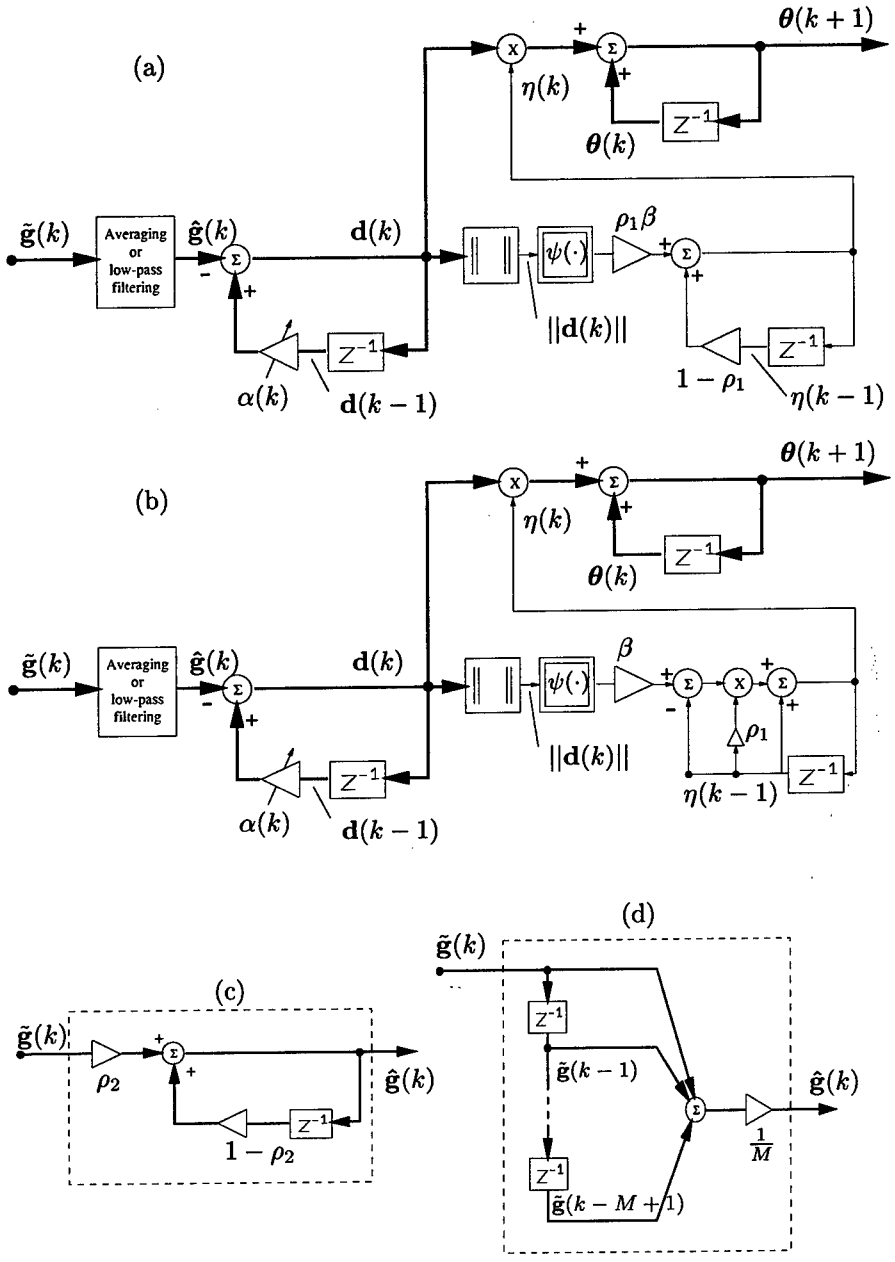


Figure 1: Functional block diagram illustrating hardware implementation of the new learning schemas: (a) Eqs. (13)-(15); (b) Eqs (13),(14),(16). (c) IIR first-order low-pass filter; (d) Standard averaging with sliding window, with null initial conditions. For $\alpha(k) = 0$, the learning rules simplify to Eqs (3)-(6).

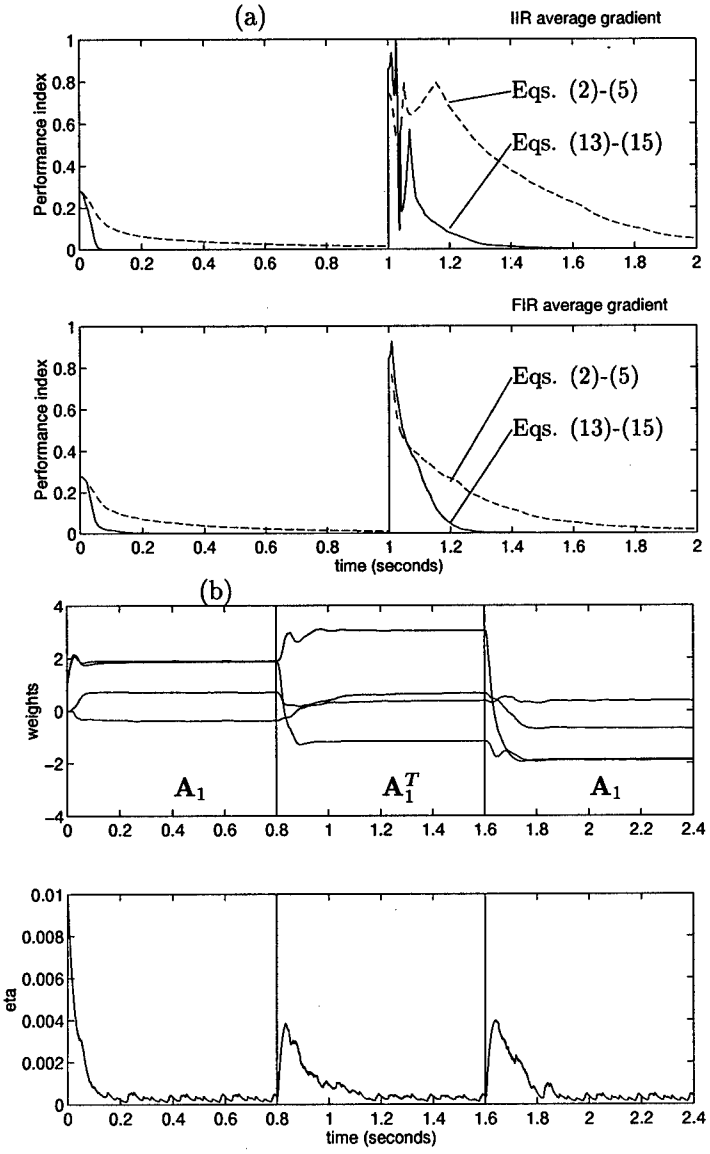


Figure 2: (a) Comparison of performance indexes obtained with gradient descent (Eqs. (2)-(5)) and conjugate gradient (Eqs. (13)-(18)). Both algorithms use the adaptive learning rate of Eq. 15, with parameters $\beta = 0.005, \eta(0) = 0.01, \rho_2 = 0.01$; restarts are performed every $R = 4$ iterations (conjugate gradient case). Upper figure: the gradient is estimated using an IIR filter of parameter $\rho_1 = \rho_2$. Lower figure: the gradient is estimated using a FIR filter of parameter $M = 100$. (b) Weights and learning rate for Eqs. (13)-(15) and IIR filter, in case of a double change of mixing matrix.

and with matrix $\mathbf{A}_2 = \mathbf{A}_1^T$ during the next second. Second experiment (see fig. 2 (b)): mixing with $\mathbf{A}_1$ during 0.8s, $\mathbf{A}_2 = \mathbf{A}_1^T$ during the next 0.8s, and again with $\mathbf{A}_1$. In order to assess the performance of our algorithm we use the following normalized performance index PI, which provides a measure of crosstalking or estimation of the closeness between $\mathbf{W}^{-1}$ and the desired mixing matrix $\mathbf{A}$ while taking into consideration the indeterminacy (scaling and order of the estimated sources) inherent to the blind separation problem [4, 8]:

$$PI(k) = \frac{1}{m}\sum_{i=1}^{m}\left\{\sum_{j=1}^{m}\frac{|p_{ij}|^2}{\max_q |p_{iq}|^2} - 1\right\} + \frac{1}{m}\sum_{j=1}^{m}\left\{\sum_{i=1}^{m}\frac{|p_{ij}|^2}{\max_q |p_{qj}|^2} - 1\right\}, \tag{21}$$

where $\mathbf{P}(k) = [p_{ij}] = \mathbf{W}(k)\mathbf{A}$.

Computer simulation results show that the proposed algorithm has good dynamic convergence and tracking capabilities. On basis of intensive computer simulation experiments we have found that the new algorithm with Fletcher - Reeves formula with resetting of $\alpha(k)$ every $R \cong n$ ($n = m^2$, number of parameters) iterations or if $\alpha(k)$ achieves a value greater than 1, provided better performance as compared to Polak-Ribiere formula or algorithms given by Eqs. (2)-(6). Furthermore, averaging with sliding time window by M from 50-100 provides slightly better performance than using first order IIR low-pass filter.

5 CONCLUSIONS

A new class of on-line adaptive learning algorithms with a variable step size is proposed. In our algorithms the learning rate is adjusted depending on the value of gradient norm or strictly speaking low-pass filtered version of the norm of actual search direction. The low-pass filtering process is optimized by employing a conjugate gradient approach. In contrast to other conjugate gradient approaches which require knowledge of a cost function, our algorithms can be applied either when a cost function is available (supervised case) or not (unsupervised case). In this paper we have presented an application to the problem of blind separation of sources illustrating the applicability of the algorithm to the unsupervised case.
The main advantages are that the proposed algorithm adapts with high convergence speed to a fast or slow change of the system and also produces a small final misadjustment error. This allows the adaptive system to track slow changes or to adapt relatively quickly to abrupt changes, as well as, to produce a small steady state misadjustment.

References

[1] S. Amari (1967), A theory of adaptive pattern classifiers, IEEE Trans. on Electronic Computers, vol. EC-16, No.3, pp.299-307.

[2] S. Amari (1997), Natural gradient works efficiently in learning, Neural Computation (submitted).

[3] S. Amari, S.C. Douglas, A. Cichocki and H.H. Yang (1997), Multichannel blind deconvolution and equalization using the natural gradient, In Proc. of *Signal Processing Advance in Wireless Communication Workshop*, Paris, pp. 101-104.

[4] S. Amari, A.Cichocki and H. Yang (1996), A new learning algorithm for blind signal separation, *Advances in Neural Information Processing Systems (1995)*, **8**, Eds. David S. Touretzky, Michakel C. Mozer and Michael E. Hasselmo, MIT Press: Cambridge, MA, pp. 757-763.

[5] A. J. Bell and T. J. Sejnowski (1995), An information-maximization approach to blind separation and blind deconvolution, *Neural Computation*, **7**, pp. 1129-1159.

[6] G.K. Boray and M.D. Srinath (1992), Conjugate gradient techniques for adaptive filtering. *IEEE Transactions on Circuits and Systems*, 39(1):1–10.

[7] J.F. Cardoso and B. Laheld (1996), Equivariant adaptive source separation, *IEEE Trans. on Signal Processing*, vol. 44, pp 3017-3030.

[8] A. Cichocki, S. Amari, M. Adachi, and W. Kasprzak (1996), Self-adaptive neural networks for blind separation of sources, *Proceedings of Int. Symp. on Circuits and Systems (ISCAS-96)*, May 1996, Atlanta, GA, USA, vol. 2, pp. 157–161.

[9] A. Cichocki, S. Amari and J. Cao (1996), Blind separation of delayed and convolved signals with self-adaptive learning rate. In Proc. of Int. Symposium on Non-Linear Theory and its Applications (NOLTA'96), Kochi, Japan, Oct. 7-9, pp. 229-232.

[10] R. Fletcher (1987), *Practical Methods of optimization.* Wiley.

[11] V.J. Mathews and Z. Xie (1993),A stochastic gradient adaptive filter with gradient adaptive step size, IEEE Trans. on Signal Processing, vol. 41, pp.2075-2087.

[12] N. Murata, K. Mueller, A. Ziehle and S-I. Amari (1996), Adaptive on-line learning in changing environments, *in NIPS-96, MIT Press, vol.9* (in print).

[13] H. Sompolinsky, N. Barkai and H.S. Seung (1995), On-line learning of dichotomies: algorithms and learning curves, *in Neural Networks : The Statistical Mechanics Perspective*, Eds J-h. Oh, C. Kwon, and S. Cho, World Scientific, Singapore 1995, pp. 105-130.

[14] J. Principe, C. Wang and H. Wu (1996), Temporal decorrelation using teacher forcing anti-Hebbian learning and its applications in adaptive blind source separation. In Proc. IEEE Workshop on Neural Networks for Signal Processing, Kyoto, Japan, pp. 413-422.

[15] F.-B. Ueng, Y. T. Su (1997), Adaptive VSS blind equalizers. IEEE Signal Processing Letters vol. 4, pp. 100-102.

Segmentation and Identification of Drifting Dynamical Systems

J. Kohlmorgen[1], K.-R. Müller[1], K. Pawelzik[2]

[1] GMD FIRST, Rudower Chaussee 5, D-12489 Berlin, Germany
e-mail: {jek, klaus}@first.gmd.de
[2] MPI für Strömungsforschung and SFB 185: Nonlinear Dynamics
Bunsenstr. 10, D-37073 Göttingen, Germany
e-mail: klaus@chaos.uni-frankfurt.de

Abstract. A method for the analysis of nonstationary time series with multiple operating modes is presented. In particular, it is possible to detect and to model a switching of the dynamics and also a less abrupt, time consuming drift from one mode to another. This is achieved by an unsupervised algorithm that segments the data according to inherent modes, and a subsequent search through the space of possible drifts. An application to physiological wake/sleep data demonstrates that analysis and modeling of real-world time series can be improved when the drift paradigm is taken into account. In the case of wake/sleep data, we hope to gain more insight into the physiological processes that are involved in the transition from wake to sleep.

1 Introduction

Modeling dynamical systems through a measured time series is commonly done by reconstructing the state space with time-delay coordinates [11, 16]. The prediction of the time series can then be accomplished by training neural networks [17]. If, however, a system operates in multiple modes and the dynamics is *drifting* or *switching*, standard approaches like multi-layer perceptrons are likely to fail to represent the underlying input-output relations. Moreover, they do not reveal the dynamical structure of the system. Time series from alternating dynamics can originate from many kinds of systems in physics, biology and engineering. Phenomena of this kind are observed, for example, in speech [15], brain data [10, 12], or dynamical systems which switch attractors [3].

In [5, 9, 13], we have described a framework for time series from *switching* dynamics, in which an ensemble of neural network predictors specializes on the respective operating modes. Related approaches can be found in [1, 18]. We now extend the ability to describe a mode change not only as a switching but – if appropriate – also as a drift from one predictor to another. Our results indicate that physiological signals contain drifting dynamics, which underlines the potential relevance of our method in time series analysis.

0-7803-4256-9/97/$10.00

2 Detecting Drifts in the Dynamics

The detection and analysis of drifts is performed in two steps. First, an unsupervised (hard-)segmentation method is applied. It was first presented in [4]. In this approach, an ensemble of prediction experts f_i, $i = 1, ..., N$, is trained by maximizing the likelihood that the ensemble would have generated the time series. For the derivative of the log-likelihood with respect to the output of an expert, we get (cf. [5])

$$\frac{\partial \log L}{\partial f_i} \propto \left[\frac{e^{-\beta(y-f_i)^2}}{\sum_j e^{-\beta(y-f_j)^2}} \right] (y - f_i), \tag{1}$$

where y is a data point to be predicted. This learning rule can be interpreted as a weighting of the learning rate of each expert by the expert's relative prediction performance. It is a special case of the Mixtures of Experts [2] learning rule, with the gating network being omitted. Furthermore, we imposed a low-pass filter on the prediction errors and used deterministic annealing in the training process (see [5, 13] for details).[3]

As a prerequisite of this method, mode changes should occur infrequent, i.e. between two mode changes the dynamics should operate stationary in one mode for a certain number of time steps. Applying this method to a time series yields a (hard) segmentation of the series into different operating modes together with prediction experts for each mode. In case of a drift between two modes, the respective segment tends to be subdivided into several parts, because a single predictor is not able to handle the nonstationarity.

The second step takes the drift into account. A segmentation algorithm is applied that allows to model drifts between two stationary modes by combining the two respective predictors, f_i and f_j. The drift is modeled by a weighted superposition

$$f(\mathbf{x}_t) \;=\; a(t)\, f_i(\mathbf{x}_t) \;+\; (1 - a(t))\, f_j(\mathbf{x}_t), \quad 0 \le a(t) \le 1, \tag{2}$$

where $a(t)$ is a mixing coefficient and $\mathbf{x}_t = (x_t, x_{t-\tau}, \ldots, x_{t-(m-1)\tau})^T$ is the vector of time-delay coordinates of a (scalar) time series $\{x_t\}$. Furthermore, m is the embedding dimension and τ is the delay parameter of the embedding. Note that the use of multivariate time series is straightforward.

The drift segmentation algorithm performs a complete search for the optimal segmentation with the lowest average prediction error and takes the described drift into account. The search is performed in the following way: For a given time series, which is not necessarily the training set used in the training phase, each of the previously trained experts performs a prediction for every time step, which results in a matrix of expert outputs $f_i(\mathbf{x}_t)$ versus time steps t. This matrix can then be used to compute the mean prediction errors for arbitrary segmentations of the time series, including drifts of any length and shape. The best segmentation with the lowest prediction error can be obtained efficiently by dynamic programming.

[3] Further information and papers can be found at:
`http://www.first.gmd.de/persons/Kohlmorgen.Jens.html`

2.1 The Drift Segmentation Algorithm in Detail

Consider a set P of 'pure' states (dynamical modes). Each state $s \in P$ represents a single neural network $k(s)$, which solely performs a prediction. Next, consider a set M of 'mixed' states, where each state $s \in M$ represents a linear mixture of two nets $i(s)$ and $j(s)$. Then, given a state $s \in S, S = P \cup M$, the prediction of the overall system is performed by

$$g_s(\mathbf{x}_t) = \begin{cases} f_{k(s)}(\mathbf{x}_t) & ; \text{if } s \in P \\ a(s) f_{i(s)}(\mathbf{x}_t) + b(s) f_{j(s)}(\mathbf{x}_t) & ; \text{if } s \in M \end{cases} \tag{3}$$

For each mixed state $s \in M$, the coefficients $a(s)$ and $b(s)$ have to be set together with the respective network indices $i(s)$ and $j(s)$. For computational feasibility, the number of mixed states has to be restricted. Our intention is to allow for drifts between any two network outputs of the previously trained ensemble. We choose $a(s)$ and $b(s)$ such that $0 < a(s) < 1$ and $b(s) = 1 - a(s)$. Moreover, the algorithm only allows for a discrete set of mixed states. Consequently, a discrete set of $a(s)$ values has to be defined. For simplicity, equally distant steps can be chosen,

$$a_r = \frac{r}{R+1}\ , \ r = 1, \ldots, R. \tag{4}$$

R is the number of intermediate mixture levels. A given resolution R between any two out of N nets yields a total number of mixed states $|M| = R \cdot N \cdot (N-1)/2$. For example, in this paper the resolution $R = 32$ is used. Assume $N = 8$, then there are $|M| = 896$ mixed states, plus $|P| = N = 8$ pure states where only single nets are considered.

A dynamic programming technique, equivalent to the Viterbi algorithm for Hidden Markov Models (HMM) [15], efficiently yields the sequence of nets and linear mixtures of nets with the lowest prediction cost C^*. C^* is the sum of squared prediction errors plus transition costs for the best fitting sequence. This sequence can be obtained between two points in time, t_0 and t_{max}, by recursively computing, for all $s \in S$, the cost $C_s(t)$ of the most likely state sequence that might have produced the time series $x_{t_0}, \ldots, x_t$, and whose state at time t is s:

$$C_s(t_0) = \varepsilon_s(t_0), \tag{5}$$

$$C_s(t) = \varepsilon_s(t) + \min_{\hat{s} \in S} \left\{ C_{\hat{s}}(t-1) + T(\hat{s}, s) \right\}, \quad t = t_0 + 1, \ldots, t_{max}, \tag{6}$$

$$C^* = \min_{s \in S} \left\{ C_s(t_{max}) \right\}, \tag{7}$$

where

$$\varepsilon_s(t) = (x_t - g_s(\mathbf{x}_{t-\tau}))^2 \tag{8}$$

is the squared prediction error of the pure or mixed network output, and $T(\hat{s}, s)$ is the transition cost to jump from state $\hat{s}$ to state s. Note that the transition costs are in analogy to the transition probabilities in HMMs, i.e. the choice of the transition matrix T determines the transition probability between any two states. In this way, a priori knowledge about the problem can be incorporated. In the following applications, either switches or smooth drifts between two nets are allowed, all other possible transitions are disabled by setting $T(\hat{s}, s) = \infty$.

The resulting segmentation sequence is obtained by backtracking through the sequence of states that make up C^* (cf. [15]). In the context of HMMs, this segmentation can be interpreted as the most likely state sequence that could have generated the given time series, in our case with the additional assumption that mode changes occur either as (smooth) drifts or as infrequent switches.

3 Applications

To illustrate the basic idea of this approach, a simple example of drifting chaotic dynamics is discussed first. It is followed by an application to a drifting system of the Mackey-Glass model of blood cell regulation. Finally, an application to real-world data is presented: EEG data of an afternoon nap of a human.

3.1 Drifting Chaos

Consider a chaotic time series $\{x_t\}$, where $x_{t+1} = f(x_t)$, Fig.1(a). Four major operating modes are established by using four different chaotic maps:

$f_1(x) = 4x(1-x)\,,\ x \in [0,1]$	(logistic map)
$f_2(x) = f_1(f_1(x))$	(double logistic map)
$f_3(x) = 2x$, if $x \in [0,.5)$ and $2(1-x)$, if $x \in [.5,1]$	(tent map)
$f_4(x) = f_3(f_3(x))$	(double tent map)

For the first 50 time steps, f_1 is applied recursively, starting with $x_0 = 0.5289$. After $t = 50$ time steps, the dynamics is drifting from f_1 to f_2 using

$$f(x_t) \;=\; (1 - a(t))\, f_1(x_t) \;+\; a(t)\, f_2(x_t), \qquad a(t) = \frac{t - t_a}{t_b - t_a}, \qquad (9)$$

with $t_a = 50$ and $t_b = 100$. The drift is linear in time and takes another 50 time steps. Then, the system runs stationary in mode f_2 for the following 50 time steps, whereupon it is drifting to f_3 in the same fashion as before, and so on. At $t = 350$, the system starts to drift back from f_4 to f_1 and the cycle starts again at $t = 400$.

In the resulting time series one cannot determine the appropriate continuation x_{t+1}, given only x_t and no information about the operating mode. In

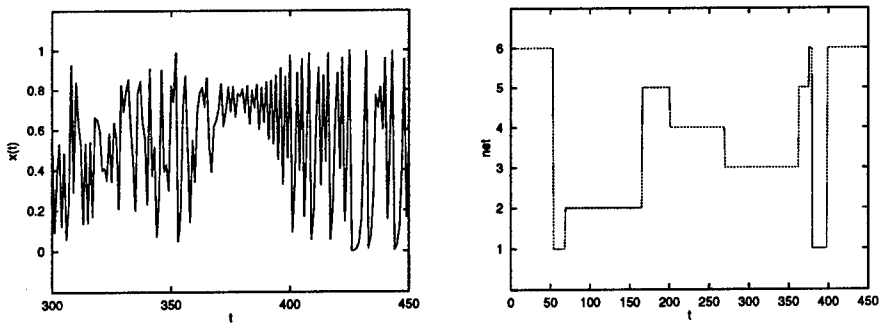


Fig. 1. *(a) A part of the training data, generated by the chaotic return maps f_1 and f_4. First, f_4 is iterated from $t = 300$ to $t = 350$. Then, there is a drift to f_1 between $t = 350$ and $t = 400$. After $t = 400$, f_1 is iterated. (b) The final segmentation into training subsets, obtained by the competitive training procedure. Shown are the first 450 data points. This segmentation cannot represent the drift. The stationary parts, f_1 in $[0, 50]$ and $[400, 450]$, f_2 in $[100, 150]$, f_3 in $[200, 250]$, f_4 in $[300, 350]$, are predicted by nets 6, 2, 4, and 3, respectively. The nonstationary drift parts in between are shared among all predictors, including nets 1 and 5.*

principle, one way to solve this problem is the method of time-delay embedding [11]. In this case, however, the inclusion of such kind of memory, e.g. using also x_{t-1}, leads to a very complex prediction function [14]. Moreover, using a single network for prediction does not reveal the dynamical structure of the system. An adequate representation of the underlying relations should therefore contain a division into subtasks, as it is performed within our framework.

First, the competing experts approach [5, 9, 13] is applied to the first 1200 data points of the generated time series. An ensemble of 6 predictors $\tilde{f}_i(x_t)$, $i = 1, ..., 6$, competes for the data during the training phase. We use radial basis function (RBF) networks of the Moody-Darken type [8] as predictors, because they offer a fast and robust learning method. Each predictor has 20 basis functions. After training, four predictors have specialized each on a different chaotic map, and the other two predictors tried to specialize on the drift parts. This can be observed in the final segmentation of the competition procedure, shown in Fig.1(b).

Next, the drift segmentation algorithm is applied to all six networks. It perfectly reproduces the behavior of the dynamics, as seen in Fig.2(a) for the resolution $R = 32$: a linear drift between four stationary operating modes is obtained. Fig.2(b) is included to demonstrate the effect of a lower resolution.

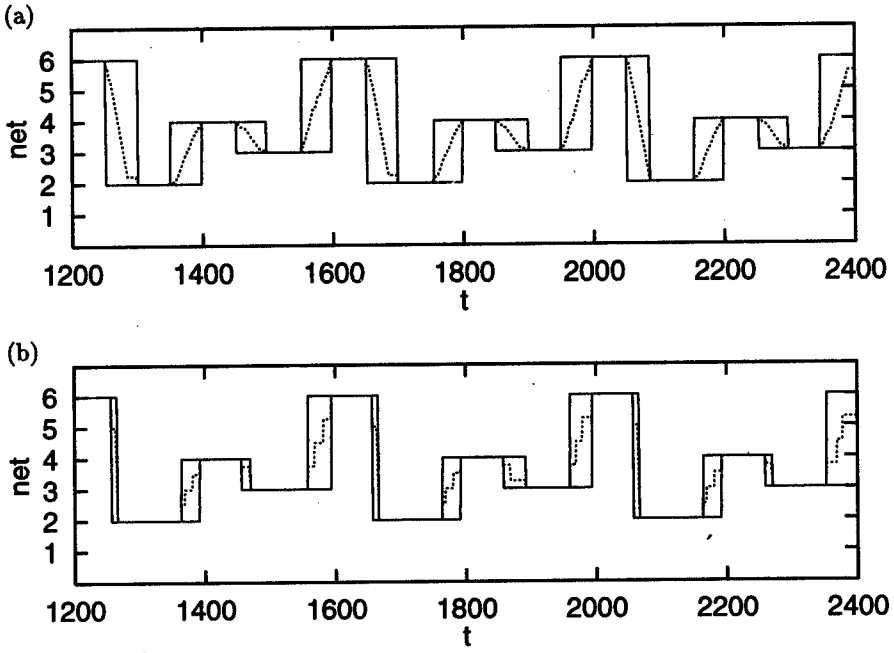


Fig. 2. *(a) The segmentation obtained by the drift algorithm on the test data* $[1200, 2400]$*, using the resolution* $R = 32$*. Shown is the sequence of nets as a function of time. The dotted line indicates the evolution of the mixing coefficient* $a(t)$ *of the respective nets. For example, between* $t = 1350$ *and* 1400 *it denotes a drift from net* 2 *to net* 4*, which in this case turns out to be a linear drift, as expected. The segmentation almost perfectly reproduces the behavior of the dynamical system. (b) A segmentation with a low resolution,* $R = 3$*, traverses the drift parts in* 3 *steps.*

3.2 A Drifting Mackey-Glass System

Consider a high-dimensional chaotic system generated by the Mackey-Glass delay differential equation

$$\frac{dx(t)}{dt} = -0.1x(t) + \frac{0.2x(t-t_d)}{1+x(t-t_d)^{10}}. \tag{10}$$

It was originally introduced as a model of blood cell regulation [7]. Two stationary operating modes, A and B, are established by using different delays, $t_d = 17$ and 23, respectively. After operating 100 time steps in mode A (with respect to a subsampling step size $\tau = 6$), the dynamics is drifting to mode B. The drift takes another 100 time steps. It is performed by mixing the equations for $t_d = 17$ and 23 during the integration of eq.(10). The mixture is generated according to eq.(2), using an exponential drift

$$a(t) = \exp\left(\frac{-4t}{100}\right), \quad t = 1, \ldots, 100. \tag{11}$$

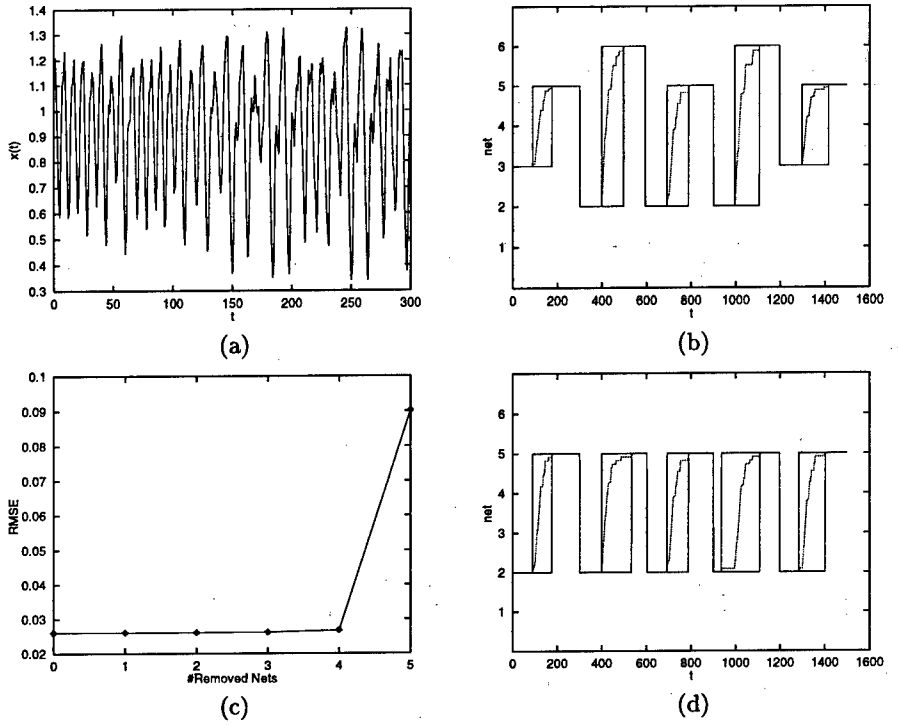


Fig. 3. *(a) The drifting Mackey-Glass time series. The dynamical system operates in mode A for the first 100 time steps. Then, the dynamics is drifting to mode B during the next 100 steps, and remains stationary in B. After $t = 300$, the system switches back to mode A and the cycle starts again. (b) The resulting drift segmentation invokes four nets. This is because two nets became experts for mode A, and two others for mode B. (c) Increase of the prediction error when predictors are successively removed. Although no further training has been performed, up to four predictors can be removed without a significant increase of the prediction error. (d) The two remaining predictors model the dynamics of the time series properly.*

Then, the system runs stationary in mode B for the following 100 time steps, whereupon it is *switching* back to mode A at $t = 300$, and the loop starts again (Fig.3(a)). The competing experts algorithm is applied to the first 1500 data points of the generated time series, using an ensemble of 6 predictors $\tilde{f}_i(\mathbf{x}_t)$, $i = 1, ..., 6$. The input to each predictor is a vector $\mathbf{x}_t$ of time-delay coordinates of the scalar time series $\{x_t\}$. The embedding dimension is $m = 6$ and the delay parameter is $\tau = 1$ on the subsampled data. The RBF predictors consist of 40 basis functions each.

After training, nets 2 and 3 have specialized on mode A, nets 5 and 6 on mode B. This can be seen in the drift segmentation in Fig.3(b). Moreover, the removal of four nets does not increase the root mean squared error (RMSE)

of the prediction significantly (Fig.3(c)), which correctly indicates that two predictors completely describe the dynamical system. The sequence of nets to be removed is obtained by repeatedly computing the RMSE of all n subsets with $n-1$ nets each, and selecting the subset with the lowest RMSE of the respective drift segmentation. The segmentation of the remaining nets, 2 and 5, nicely reproduces the evolution of the dynamics, as seen in Fig.3(d).

3.3 Wake/Sleep Data

In [10], we analyzed physiological data recorded from the wake/sleep transition of a human. The objective was to provide an unsupervised method to detect the sleep onset and to give a detailed approximation of the signal dynamics with a high time resolution, ultimately to be used in diagnosis and treatment of sleep disorders. The application of the drift segmentation algorithm now yields a more detailed modeling of the dynamical system.

As an example, Fig. 4 shows a comparison of drift segmentation ($R = 32$), hard segmentation ($R = 0$, i.e. no drift), and a manual segmentation by a medical expert. The experimental data was measured during an afternoon nap of a healthy human. The computer-based analysis is performed on a single-channel EEG recording (occipital-1), whereas the manual segmentation was worked out using six physiological signals (EEG, EOG, ECG, heart rate, blood pressure, respiration).

The drift algorithm yields several drift parts. The sleep onset, according to the manual segmentation at $t \approx 4000$, is represented by an exponential drift from a wake-state predictor, net 7, to a sleep-state predictor, net 4. On the other hand, the wake-up is introduced at $t \approx 9000$ by a slight drift back to net 7, which holds until the wake-up point is reached ($t \approx 9500$ in the manual segmentation). There, a sudden change of the mixing coefficient gives more weight to wake-state net 7. After $t \approx 9800$ (eyes open), a mixture of two wake-state nets, 2 and 7, performs the prediction.

Compared to both hard segmentations, the drift segmentation reveals several interesting details of the dynamical changes in the transition between different wake/sleep stages. A more comprehensive analysis of the wake/sleep data is beyond the scope of this contribution and we would like to refer the reader to our forthcoming publication [6].

4 Summary and Discussion

A method for the unsupervised segmentation and identification of nonstationary drifting dynamics was presented. It applies to time series where the dynamics drifts or switches between different operating modes. The method was illustrated in two cases of drifting chaotic systems. An application to physiological wake/sleep data demonstrates that drift can be found in natural systems. It is therefore important to consider this aspect of data description.

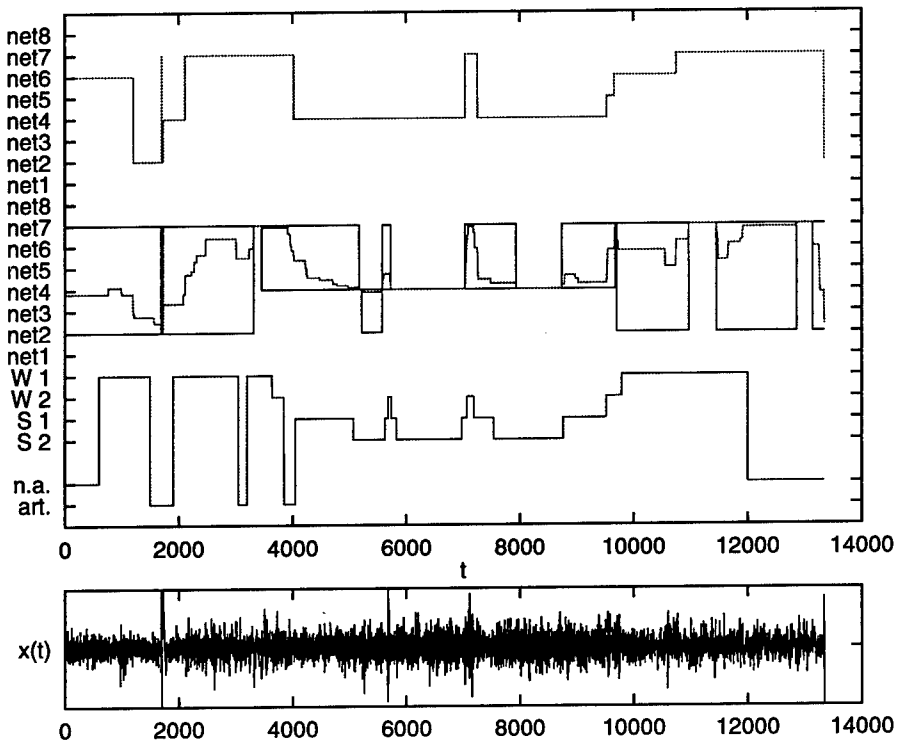


Fig. 4. *Comparison of hard segmentation (upper), drift segmentation (middle), and a manual segmentation by a medical expert (lower). Only a single-channel EEG recording (occipital-1,* 1400 *sec.) of an afternoon nap is given for the two algorithmic approaches.* $W1$ *and* $W2$ *indicate two wake-states (eyes open/closed) in the manual analysis,* $S1$ *and* $S2$ *indicate sleep stage I and II, respectively (*n.a.*: not considered,* art.*: artifacts). Compared to the manual segmentation, the EEG is properly segmented by our method.*

In the case of wake/sleep data, where the physiological state transitions are far from being understood, we are able to extract the shape of the dynamical drift from wake to sleep in an unsupervised manner. By applying this new analysis tool, we hope to gain more insights into the underlying physiological processes. Our future work is therefore dedicated to a comprehensive analysis of large physiological datasets. We expect, however, that our method will be also applicable in many other fields.

Acknowledgments

We acknowledge support of the DFG (grant Ja379/51). Furthermore, we would like to thank J. Rittweger for the physiological data and for fruitful discussions.

References

1. Cacciatore, T.W., Nowlan, S.J. (1994). Mixtures of Controllers for Jump Linear and Non–linear Plants. NIPS'93, Morgan Kaufmann.
2. Jacobs, R.A., Jordan, M.A., Nowlan, S.J., Hinton, G.E. (1991). Adaptive Mixtures of Local Experts, *Neural Computation* **3**, 79–87.
3. Kaneko, K. (1989). Chaotic but Regular Posi-Nega Switch among Coded Attractors by Cluster-Size Variation, Phys. Rev. Letters **63**, 219.
4. Kohlmorgen, J., Müller, K.-R., Pawelzik, K. (1994). Competing Predictors Segment and Identify Switching Dynamics. Proceedings of ICANN'94, Springer London, 1045–1048.
5. Kohlmorgen, J., Müller, K.-R., Pawelzik, K. (1995). Improving short-term prediction with competing experts. Proceedings of ICANN'95, EC2 & Cie, Paris, 2:215–220.
6. Kohlmorgen, J., Müller, K.-R., Rittweger, J., Pawelzik, K., in preparation.
7. Mackey, M., Glass, L. (1977). Oscillation and Chaos in a Physiological Control System, Science **197**, 287.
8. Moody, J., C. Darken (1988). Learning with Localized Receptive Fields. In *Proceedings of the 1988 Connectionist Models Summer School* (Pittsburg 1988), eds. D. Touretzky, G. Hinton, and T. Sejnowski, 133–143. San Mateo: Morgan Kaufmann.
9. Müller, K.-R., Kohlmorgen, J., Pawelzik, K. (1995). Analysis of Switching Dynamics with Competing Neural Networks, *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E78-A, No.10, 1306–1315.
10. Müller, K.-R., Kohlmorgen, J., Rittweger, J., Pawelzik, K. (1995). Analysing Physiological Data from the Wake-Sleep State Transition with Competing Predictors, in NOLTA'95: Las Vegas Symposium on Nonlinear Theory and its Applications, 223–226.
11. Packard, N.H., Crutchfield J.P., Farmer, J.D., Shaw, R.S. (1980). Geometry from a Time Series. *Physical Review Letters*, 45:712–716.
12. Pawelzik, K. (1994). Detecting coherence in neuronal data. In: Domany, E., Van Hemmen, L., Schulten, K., (Eds.), *Physics of neural networks*, Springer.
13. Pawelzik, K., Kohlmorgen, J., Müller, K.-R. (1996). Annealed Competition of Experts for a Segmentation and Classification of Switching Dynamics, *Neural Computation*, **8:2**, 342–358.
14. Pawelzik, K., Müller, K.-R., Kohlmorgen, J. (1997). Divisive Strategies for Predicting Non-Autonomous and Mixed Systems, submitted to *Neural Processing Letters*.
15. Rabiner, L.R. (1988). A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. In *Readings in Speech Recognition*, ed. A. Waibel, K. Lee, 267–296. San Mateo: Morgan Kaufmann, 1990.
16. Takens, F. (1981). Detecting Strange Attractors in Turbulence. In: Rand, D., Young, L.-S., (Eds.), *Dynamical Systems and Turbulence*, Springer Lecture Notes in Mathematics, **898**, 366.
17. Weigend, A.S., Gershenfeld, N.A. (Eds.) (1994). *Time Series Prediction: Forecasting the Future and Understanding the Past*, Addison-Wesley.
18. Weigend, A.S., Mangeas, M. (1995). Nonlinear gated experts for time series: discovering regimes and avoiding overfitting, *International Journal of Neural Systems* **6**, 373–399.

AN IMPROVED SCHEME FOR THE FUZZIFIER IN FUZZY CLUSTERING.

L. ben Romdhane, B. Ayeb & S. Wang
Mathematical and Computer Science Department
Faculty of Sciences, University Of Sherbrooke
Sherbrooke, Québec, Canada J1K 2R1

ABSTRACT

Clustering is an important research area and of practical applications in many fields. Fuzzy clustering has shown advantages over crisp and probabilistic clustering especially when there are *significant* overlaps between clusters [1], [2], [7], [9], [10]. However, all of the fuzzy clustering algorithms are sensitive to an *exponent* parameter, namely the fuzzifier. To our knowledge, no theoretical foundations are yet available for the optimal choice of this parameter [2], [3], [4], [5], [6], [9], [10]. The current work develops an improved scheme for the fuzzifier by embedding more knowledge about the data set to cluster in its computation.

Keywords: Neural Networks, Clustering, Fuzzy Set Memberships, Fuzzifier.

1 INTRODUCTION

Clustering is an important research area and of practical applications in a variety of fields, including pattern recognition and data compression [11]. Clustering algorithms attempt to partition the input data into groups, i.e. clusters, such that patterns within a cluster are similar to each other than are patterns in distinct clusters [5]. Fuzzy clustering algorithms have shown advantages over their Crisp / Probabilistic counterparts especially when there are significant overlaps between clusters. Investigation of the motivations for

0-7803-4256-9/97/$10.00

introducing the fuzzy set membership notion within clustering algorithms goes beyond the scope of the current paper. For a concise review of fuzzy clustering schemes and an experiment-based comparison with their Crisp / Probabilistic counterparts we refer the reader to the literature; e.g. [3], [4], [5], [9], [10], [12].

By adopting the fuzzy set membership notion, many fuzzy clustering algorithms were proposed; e.g. the Fuzzy Learning Vector Quantization (FLVQ, for short) [2], the Fuzzy C-Means (FCM, for short) [5], the Fuzzy C Spherical Shells (FCSS, for short) [10]. A Fuzzy clustering algorithm makes use of an exponent parameter said to be the *fuzzifier*. It constitutes the most problematical choice for fuzzy clustering. Indeed, this parameter affects the *convergence rate* as well as the *cluster validity* of the algorithm. To our knowledge no theoretical foundations for an optimal choice of this parameter are yet available [2], [3], [4], [5] [6], [9], [10]. Instead, an adequate choice is always done via experimentation [3], [4], [5] [6], [9], [10], i.e. this choice is still largely *heuristic*.

A general rule of thumb about the *setting* of the fuzzifier was proposed and analyzed in literature [4], [5], [2]. In the present paper, we propose a solution that makes use of this general rule of thumb. Besides, we make closer interactions between the fuzzifier and the data set to cluster.

The remainder of this paper is organized as follows. In Section 2 we formulate the clustering problem and give the main notations adopted in this paper. In section 3 we discuss the problems faced by an inadequate choice of the fuzzifier and outline our proposal and related work. In section 4 preliminary experimental results are reported. The final section offers discussions and conclusions.

2 NOTATION AND PROBLEM FORMULATION

Let c be an integer, $1 < c < n$ and let $X = \{x_1, x_2, \cdots, x_n\}$ be a set of n feature vectors in $\mathcal{R}^p$. $\mathbf{X}$ is a numerical object data. x_j is a representation of the jth object in $\mathbf{X}$. x_{jk} is the kth feature value of the jth object. The number of clusters can be specified *a priori* or computed from the input data set using criteria of optimality such that the fuzzy hypervolume and density [6]. A cluster is defined by computing its centroïd; i.e. class prototype. For this let $V = \{\nu_1, \nu_2, \cdots, \nu_c\}$ be the set of centroïds, $\nu_i \in \mathcal{R}^p (1 \leq i \leq c)$.

Fuzzy clustering algorithms find the optimal partition of the input data into clusters by minimizing an error criterion, namely the *weighted within groups sum of squared errors objective function* [2], [3], [4], [5], [6], [9]:

$$J_m(U, V; X) = \sum_{k=1}^{n} \sum_{i=1}^{c} (u_{ik})^m \parallel x_k - \nu_i \parallel_A^2 \qquad (1)$$

subject to the constraints:

$$0 < \sum_{k=1}^{n} u_{ik} < n \quad \forall i$$

$$\sum_{i=1}^{c} u_{ik} = 1 \quad \forall i,k \tag{2}$$

where u_{ik} is the membership degree of the kth feature vector in the ith cluster, U is a $c \times n$ matrix of u_{ik} values ($1 \leq i \leq c,\ 1 \leq k \leq n$); and $\| \ \|_A^2$ is a distance measure w.r.t. the matrix A. U is called the *fuzzy c-partition* of the initial data set. m is the *fuzzifier* used by any fuzzy clustering algorithm.

The following theorem [2] states conditions on the fuzzy c-partition in order to minimize the objective function:

Theorem 1 *Assume* $\| x_k - \nu_j \|_A^2 > 0, \forall j,k$. *(U,V) may minimize* J_m *only if, for* $m > 1$*:*

$$u_{ik} = \left(\sum_{j=1}^{c} \left(\frac{\| x_k - \nu_i \|_A}{\| x_k - \nu_j \|_A} \right)^{\frac{2}{m-1}} \right)^{-1} \quad \forall i,k \tag{3}$$

$$\nu_i = \frac{\sum_{k=1}^{n} (u_{i,k})^m x_k}{\sum_{k=1}^{n} (u_{i,k})^m} \quad \forall i \tag{4}$$

The following section discusses the problems encountered by an inadequate choice of this parameter and introduces a general rule of thumb about its setting.

3 AN IMPROVED SCHEME FOR THE FUZZIFIER

3.1 Basics

The fuzzifier is used to *weight* the distances between the prototype vectors and the input vectors in the computation of the membership values. The fuzzifier controls the fuzziness of the c-partition to be computed [4], [5]. This "amount" of fuzziness ranges from absolute hard clustering (at $m = 1$) to increasingly fuzzy clustering as the fuzzifier takes larger values. This means that the discrepancy between memberships of a pattern in different clusters is emphasized (resp. reduced) by adopting low values (resp. large values) for the fuzzifier. Furthermore, this parameter influences considerably the

convergence rate of the considered algorithms and the *cluster validity* of the data set at hand.

To our knowledge, no theoretical foundations for the optimal choice of this parameter are yet available [3], [5], [4], [6], [2], [9], [10]. In the lack of theoretical basis for the optimal fuzzifier that produces best clustering, the *appropriate* value of this parameter is set via experimentation [3], [5], [4], [6], [9], [10]. In essence, the choice is still largely heuristic.

But, one can point out a general rule of thumb that guides the changing of this parameter and that addresses both the convergence rate and the cluster validity. It can be paraphrased as follows. In fact, clustering is considered as a process of competition between prototypes. A simple analysis of (3) and (4) reveals the following remarks. By adopting large values for the fuzzifier, *each* prototype will be updated to almost the *same small* rate since:

$$\lim_{m \to \infty} u_{i,k}(t) = \frac{1}{c} \tag{5}$$

And therefore distant prototypes will win as much as closer ones. Consequently, no prototype is left during the competition process. However, large values are not beneficial to the convergence rate. Consequently, by decreasing the fuzzifier *gradually* to low values, the convergence rate will be improved.

As a conclusion to this analysis a *decreasing* scheme is more appropriate. Earlier works about this subject are reported subsequently.

3.2 Related Work

In works reported in [5] and [2], the fuzzifier was allowed to decrease *linearly* through iterations from a high value to a low terminating value meeting the requirements discussed in §3.1. In [2], this solution was used in the Descending FLVQ (↓FLVQ, for short). ↓FLVQ has a superior classification rate than FCM [4]. Although the adopted decreasing schemes for the fuzzifier in [2] and [5] improve the cluster validity and the convergence rate, they are still very sensitive to the initial and final values of the fuzzifier.

It is reported in literature that an appropriate specification of the fuzzifier requires knowledge of the characteristics of the data set at hand [4]. Hence, creating closer interactions between the fuzzifier and the input data would perform better results. It is this property that we investigate and use in our solution discussed in the next section.

3.3 Our Proposal

↓FLVQ [2] adopts an interesting approach to the problem of choosing the fuzzifier. However, in ↓FLVQ the fuzzifier takes the *same value* for all input patterns in each iterations which may be non optimal. Moreover, the ↓FLVQ is very sensitive to the *final* and *initial* values of the fuzzifier [2]. One can notice the fact that the boundaries between clusters might be fuzzy. In such

case clusters are said to *overlap* [6]. The degree of overlapping is not the same for all clusters. Consequently, they should not be treated equally as in ↓FLVQ [2]. In fact, the less the overlapping between clusters is, the lower the values of the fuzzifier should be. Consequently, this allows us to better control the "fuzziness" ("hardness") of each cluster by taking into consideration the data set at hand.

To make the fuzzifier depend on the data set to cluster, we have investigated the possibility of using the computed memberships of an input vector in all possible classes. In fact, the more these memberships are away from each other, the more a vector is attracted towards a particular class. In a formal manner, we propose the following scheme:

$$m_k(t) = m_0 \gamma_c \left(1 - \beta_k(t)\right) + \delta \quad \forall k = 1 \cdots n \tag{6}$$

$$\gamma_c = \frac{c}{c-1} \tag{7}$$

$$\beta_k(t) = \sum_{i=1}^{c} \left(u_{ik}(t)\right)^2 \qquad \forall k = 1 \cdots n \tag{8}$$

where m_0 is the initial fuzzifier, δ is a small real number ($\delta > 1.0$) and t is iteration counter.

In order to meet the requirements outlined in §3.1 the parameter $\beta_{k,t}$ should increase with time. Indeed, the computed memberships of a given pattern are more and more away from each other and therefore will have a large discrepancy since the considered pattern will be attracted through iteration to a special cluster (resp. to more than one cluster when overlapping is very important, i.e memberships are close). Due to the constraint in (2), there exist some memberships that should increase due to the decrease of the others. Thus when $\beta_k(t)$ increases, the fuzzifier decreases accordingly showing a behavior analogous to simulated annealing. Due to the constraints in (2), a simple analysis of (8) reveals that the parameter $\beta_k(t)$ ranges over $[\frac{1}{c}, 1]$. In fact, the lowest value for $\beta_k(t)$ is obtained when all the memberships are equal to $\frac{1}{c}$. The largest value for $\beta_k(t)$ is obtained when a membership degree of pattern k is 1 in one cluster and 0 in the others. Hence, from (6) the fuzzifier ranges over $[\delta, m_0 + \delta]$. This well justifies the fact that $\delta > 1$ due to **Theorem 1**, but this parameter should remain small.

Simple runs of our solution and its comparison to that adopted in the ↓FLVQ [2] are outlined in the next section.

4 PRELIMINARY EXPERIMENTS

We have used IRIS data of Anderson and Fisher [1]. It consists of three subspecies, each containing 50 samples. IRIS data is the most widely used in the experimentation of clustering algorithms [6], [5], [2], [8]. Indeed IRIS data has been widely used by researchers in clustering since 1936 [8] mainly because it presents important overlaps between subspecies (subgroups).

Unlike the ↓FLVQ [2], in our solution the fuzzifier does not take the same value for all input vectors. Hence, (3) and (4) will change respectively to:

$$u_{ik} = \left(\sum_{j=1}^{c} \left(\frac{\| x_k - \nu_i \|_A}{\| x_k - \nu_j \|_A} \right)^{\frac{2}{m_k - 1}} \right)^{-1} \quad \forall i, k; \tag{9}$$

$$\nu_i = \frac{\sum_{k=1}^{n} (u_{i,k})^{m_k} x_k}{\sum_{k=1}^{n} (u_{i,k})^{m_k}} \quad \forall i. \tag{10}$$

where m_k is computed by (6). For the sake of clarity, we note by the Data Dependent ↓FLVQ (DD-↓FLVQ, for short) the ↓FLVQ based on (9) and (10).

For a fare comparison between DD-↓FLVQ and ↓FLVQ, we have used the same parameters. We based our comparison upon two criteria: the number of steps needed by the algorithm to converge and the misclassification rate. Indeed, these two criteria are the mostly used ones for a fine comparison between competing designs [2], [5]. For the sake of simplicity, the matrix A used by (9) was set to the *identity* for both algorithms.

An initial tracking of class prototypes can be achieved using many techniques; e.g. unsupervised learning [6], a random process [5] [10]. For our concern, we will adopt the same method used in [2]. Due to the space limit, this initialization process will not be described herein. The *initial* and the *real* prototypes of the IRIS subspecies are depicted in TABLES 1 and 2 respectively.

$\nu_{1,0}^T$	4.30	2.00	1.00	1.00
$\nu_{2,0}^T$	6.10	3.20	3.95	1.30
$\nu_{3,0}^T$	7.90	4.40	6.90	2.50

TABLE 1: INITIAL GUESS OF CLASS PROTOTYPES.

Simple runs of both algorithms with different settings of their parameters are depicted in TABLE 3 where the " $\star$" means that the algorithm does not converge within the maximal maximal number of allowed iterations (= 200).

[1] IRIS data set is a courtesy of Dr James M. Keller.

ν_1^T =	5.00	3.43	1.46	0.25
ν_2^T =	5.94	2.77	4.26	1.33
ν_3^T =	6.59	2.97	5.55	2.03

TABLE 2: REAL PROTOTYPES OF THE THREE IRIS SUBSPECIES.

In order to compute the misclassification rate, the *same* training samples

Parameters	Final Prototypes						Iter.		Misc.	
	I			II			I	II	I	II
	ν_1	ν_2	ν_3	ν_1	ν_2	ν_3				
$m_0 = 8.0$	5.01	5.98	6.55	5.05	5.97	6.54				
$\epsilon = 0.01$	3.40	2.84	3.00	3.42	2.86	3.00	09	16	13	11
$\delta = 1.3$	1.50	4.44	5.36	1.45	4.42	5.36				
$\Delta m = 0.04$	0.25	1.43	1.95	0.22	1.43	1.96				
$m_0 = 10.0$	5.02	6.00	6.53	5.01	5.99	6.53				
$\epsilon = 0.01$	3.40	2.85	2.99	3.40	2.87	3.00	10	14	12	11
$\delta = 1.3$	1.50	4.45	5.32	1.49	4.46	5.32				
$\Delta m = 0.02$	0.24	1.43	1.93	0.21	1.44	1.94				
$m_0 = 7.0$				5.04	6.00	6.51				
$\epsilon = 0.001$				3.45	2.88	3.00				
$\delta = 1.1$		$\star$		1.45	4.49	5.27	$\star$	31	$\star$	13
$\Delta m = 0.03$				0.23	1.47	1.98				
$m_0 = 5.0$	5.01	5.95	6.62	4.98	5.77	6.60				
$\epsilon = 0.01$	3.40	2.82	3.02	3.37	2.84	3.02	07	16	15	12
$\delta = 1.1$	1.50	4.41	5.45	1.49	4.25	5.49				
$\Delta m = 0.01$	0.25	1.41	2.00	0.25	1.33	2.04				

TABLE 3: SIMPLE RUNS OF THE ↓FLVQ(I) AND THE DD-↓FLVQ(II) WITH VARIOUS SETTINGS OF THEIR PARAMETERS.

are classified using prototypes found by both algorithms. For this, the One-Nearest-Prototype method [2] (1-NP, for short) is adopted.

From TABLE 3, one can notice that DD-↓FLVQ is *more stable* to the changing of its parameters (ϵ, m_0, δ) than is ↓FLVQ $(\epsilon, m_0, \Delta m)$. Furthermore, from TABLES 2 and 3 one can notice that DD-↓FLVQ approximates *better* the real centroïds than does ↓FLVQ. This explains the fact that the number of misclassified patterns by DD-↓FLVQ is *less* than that by ↓FLVQ. However, DD-↓FLVQ requires in general a *greater* number of iterations to converge than ↓FLVQ.

Our experimental results [2] have shown that our solution performs better clustering than ↓FLVQ. The difference in performance is significant enough to encourage further investigations in the suggested direction.

[2]Experimental results can be provided by the authors upon request.

5 DISCUSSION AND FUTURE WORK

In a current paper, we have developed a rough solution that solves (to some extent) the problem of the choice of the fuzziness parameter in fuzzy clustering algorithms. We tried to make the fuzzifier depend on the data set to cluster. Experimental results have shown that our solution performs good clustering as that reported in [2], besides it has a *better* approximation property.

We do not pretend that our solution is better than that presented in [2] in all the cases. It may be possible to find data sets on which one solution performs better clustering than the other. Our central concern is to assess closer interactions between the fuzzifier and the input data. Although, our solution is still heuristic, the experimental results reported here show its validity and usefulness which encourage further investigations in the same direction. Embedding more knowledge about the data in the computation of the fuzzifier (e.g. cluster width, fuzzy hypervolume [6]) would perform better results. For this, an analysis of the effect of such parameters on the fuzzifier should be done. Such an analysis is not straightforward. For the time being, we are focusing our attention on the analysis of an analogous scheme to that presented in (6) embedding the fuzzy hypervolume criteria. Further studies as well as simulation results are to be included in a forthcoming paper.

ACKNOWLEDGMENTS

This research is partially supported by NSERC federal grants (OGP0121468; EQP0121469). Mr. L. Ben Romdhane is supported by ACDI fellowship.

REFERENCES

[1] J. F. Baldwin. "Fuzzy Sets, Fuzzy Clustering and Fuzzy Rules in A.I". **Lecture Notes In Artificial Intelligence**, 847:10–23, 1993.

[2] J. C. Bezdek and N. R. PAL. "Two Soft Relatives of Learning Vector Quantization". **Neural Networks**, 8(5):729–743, 1995.

[3] R. L. Cannon and J. C. Bezdek. "Efficient Implementation of the fuzzy c-means clustering algorithms". **IEEE, Transactions on Pattern Analysis and Machine Intelligence**, 8(2):248–255, 1986.

[4] F. L. Chung and T. Lee. "Fuzzy Competitive Learning". **Neural Networks**, 7(3):539–551, 1994.

[5] J. C. Bezdek E. Tsao and N. R. Pal. "Fuzzy Kohonen Clustering Networks". **Pattern Recognition**, 27(5):757–764, 1994.

[6] I. Gath and A. B. Geva. "Unsupervised Optimal Fuzzy Clustering". **IEEE, Transactions on Pattern Analysis and Machine Intelligence**, 11(7):773–781, 1989.

[7] M. M. Gupta. "Fuzzy Logic And Neural Networks". In Tenth International Conference on Multiple Criteria Decision Making(TAIPEI '92), volume 3, pages 281–294, Japan, July 19-24 1992.

[8] M. R. Gray J. M. Keller and J. A. Givens. "A Fuzzy K-Nearest Neighbor Algorithm". **IEEE, Transactions on Systems, Man and Cybernatics**, 15(4):580–585, July/August 1985.

[9] J. M. Keller and D. J. Hent. "Incorporating Fuzzy Membership Functions into the perceptron algorithm". **IEEE, Transactions on Pattern Analysis and Machine Intelligence**, 7(6):693–669, 1985.

[10] R. Krishnapuram and M. Keller. "A possibilistic Approach to Clustering". **IEEE, Transactions on Fuzzy Systems**, 1(2):98–110, May 1993.

[11] R. P. Lippman. "An Introduction to Computing with Neural Nets". **IEEE, ASSP MAGAZINE**, pages 4–22, April 1987.

[12] P. K. Simpson. **Artficial Neural Systems: Foundations, Paradigms, Apllications and implementations**. PERGAMON PRESS, 1990.

SEPARABLE NON-LINEAR LEAST-SQUARES MINIMIZATION – POSSIBLE IMPROVEMENTS FOR NEURAL NET FITTING

Jonas Sjöberg and Mats Viberg
Department of Applied Electronics, Chalmers University of Technology,
412 96 Gothenbourg, Sweden, Fax: +46-31-7721782,
Email: sjoberg@ae.chalmers.se,viberg@ae.chalmers.se

Abstract. Neural network minimization problems are often ill-conditioned and in this contribution two ways to handle this will be discussed.

It is shown that a better conditioned minimization problem can be obtained if the problem is separated with respect to the linear parameters. This will increase the convergence speed of the minimization.

The Levenberg-Marquardt minimization method is often concluded to perform better than the Gauss-Newton and the steepest descent methods on neural network minimization problems. The reason for this is investigated and it is shown that the Levenberg-Marquardt method divides the parameters into two subsets. For one subset the convergence is almost quadratic like that of the Gauss-Newton method, and on the other subset the parameters do hardly converge at all. In this way a fast convergence among the important parameters is obtained.

1. INTRODUCTION

This contribution addresses the criterion minimization to obtain the parameter estimate in a model. Two slightly different topics are covered.

It is shown that a better conditioned minimization problem can be obtained if the problem is separated with respect to the linear parameters. The parameters are divided into two sets depending on if the model is linear or nonlinear with respect to the parameter. The iterative minimization can then be done over the set of nonlinear parameters instead of over all parameters. The better conditioned problem is likely to converge faster than the original one. More aspects and an overview about separable minimization problems can be found in [5].

Many different studies have been done where different minimization algorithms are compared. See, *e.g.*, [8, 3] and further references there. In

0-7803-4256-9/97/$10.00 ©1997 IEEE

this contribution it is pointed out why the Levenberg-Marquardt algorithm often is concluded to be the best one. The reason for this is connected to the ill-conditioning, which often occurs in neural net minimization problems, see [6].

In Section 2 a short background and problem formulation is given and in Section 3 the improved conditioning of separable minimization problems is shown. Section 4 concerns the Levenberg-Marquardt algorithm and Section 5 concludes the paper.

2. PROBLEM FORMULATION

Considering the following fitting problem. Given N data $\{y(t), \varphi(t)\}_{t=1}^{N}$ which consist of an output $y(t)$ and an input regressor $\varphi(t)$. For simplicity it will be assumed that the output is one dimensional. All the results can easily be modified to cover the multi-output case.

It is assumed that there exists a function $f(\cdot)$ so that the data can be described as

$$y(t) = f(\varphi(t)) + e(t)$$

where $e(t)$ can be described by a white noise sequence. The function $f(\cdot)$ is unknown and the goal is to use the data to obtain an estimate of it. This is done by proposing a model g with adjustable parameters

$$\hat{y}(t) = g(\theta, \varphi(t)) = \sum_{k=1}^{n} c_k g_k(\varphi(t), a_k) \tag{1}$$

where $\hat{y}(t)$ is the prediction of $y(t)$ and n is the number of basis functions g_k. The parameters c_k, a_k are put into a common parameter vector $\theta = [c^T \; a^T]^T$. Depending on the choice of the basis functions $g_k(\cdot, \cdot)$, different well-known nonlinear model structures like radial basis functions and feed-forward neural nets are obtained. The dimensions of the parameter a_k also depends on the particular choice of basis functions. For example, if all basis functions are chosen as identical sigmoids, *i.e.*, $g_k(\varphi(t), a_k) = \sigma(a_{i1}^T \varphi(t) + a_{i0})$ then (1) becomes a feed-forward neural network and the number of neurons is given by n. The second index added to the parameters indicates the parameters connected to the regressor, a_{i1}, and the parameter connected to the position of the sigmoid a_{i0}.

Given a model structure g and an estimation data set $\{y(t), \varphi(t)\}_{t=1}^{N}$ the parameter estimate $\hat{\theta}$ is defined as the minimum of a criterion of fit, *e.g.*, sum of squared errors

$$\hat{\theta} = \arg\min_{\theta} V_N(\theta) \tag{2}$$

where

$$V_N(\theta) = \frac{1}{2N} \sum_{t=1}^{N} \varepsilon^2(\theta, t) \tag{3}$$

and $\varepsilon(\theta,t) = y(t) - \hat{y}(\theta,t)$.

The paper concerns possible improvements how to compute the minimum of $V_N(\theta)$, *i.e.*, to compute the parameter estimate (2). Before we come to the improvements, a more compact, vectorized, notation will be introduced which makes the calculations easier to follow.

The following notation gives a compact way to handle the whole data set at the same time $\varepsilon = [\varepsilon(\theta,1),\ \varepsilon(\theta,2),\ldots,\varepsilon(\theta,N)]^T$. Define y, $\hat{y}$, g_i, and φ in the analogous way. Introduce a matrix containing all the basis functions at all time instants $g = [g_1,\ \ldots,\ g_n]$.

Note that the model structure (1) is linear in the parameters c but non-linear in the parameters a. This feature will be exploited in the following section to derive an improved algorithm to compute the estimate $\widehat{\theta}_N$.

2.1. Iterative Minimization Scheme

Here follows a short background on minimization which forms the base for the discussions in the following two sections. For a thorough treatment of the topic see, *e.g.*, [1].

To compute $\widehat{\theta}$ (2) one typically uses an iterative gradient based algorithm of the following type

$$\widehat{\theta}^{(i+1)} = \widehat{\theta}^{(i)} - \mu_i d_i. \tag{4}$$

where μ_i is a step length to guarantee a decrease of the criterion (3) in each iteration[1]. The step direction d_i is given by

$$d_i = R_i^{-1} \nabla V_N(\widehat{\theta}_N^{(i)}) \tag{5}$$

where R_i is a matrix which modifies the search direction from the steepest descent

$$\nabla V_N(\widehat{\theta}_N^{(i)}) = \frac{1}{N} \varepsilon'^T \varepsilon \tag{6}$$

to a more favorable direction.

Starting from an initial parameter value θ_0 the equation (4) is iterated until $\widehat{\theta}^{(i)}$ converges. Depending on the choice of R_i, different minimization schemes are obtained. If R_i is chosen to the Hessian of the criterion,

$$V_N''(\widehat{\theta}_N^{(i)}) = \frac{1}{N} \varepsilon'^T \varepsilon' + \frac{1}{N} \varepsilon''^T \varepsilon \tag{7}$$

then one has the Newton method. The second term in (7) will typically be small compared to the first term and since it is much more computationally expensive to compute, it is often canceled. Also, this gives a R_i that is positive semi-definite, which is necessary for the algorithm to converge. The following alternative search directions are more common in practice:

[1]Typically one starts with $\mu_i = 1$, and test if $V_N(\widehat{\theta}^{(i+1)}) < V_N(\widehat{\theta}^{(i)})$. If that is not the case μ_i is decreased and a new $\widehat{\theta}^{(i+1)}$ is computed. This process continues until a downward step is obtained.

- *Gradient direction.* Simply take $R_i = I$.
- *Gauss-Newton direction.* Use $R_i = \frac{1}{N}\varepsilon'^T\varepsilon'$.
- *Levenberg-Marquardt direction.* Use $R_i = \frac{1}{N}\varepsilon'^T\varepsilon' + \delta_i I$. where δ_i is used instead of the step size μ_i.

For the Gauss-Newton method the computation of the step direction d_i can be formulated as linear quadratic minimization problem, which has to be solved in each iteration of (4)

$$d_i = \arg\min_{d_i} \| \varepsilon' d_i - \varepsilon \|_2 = (\varepsilon')^+ \varepsilon = (\varepsilon'^T\varepsilon')^{-1}\varepsilon'^T\varepsilon \tag{8}$$

from which the definition of the pseudo-inverse $(\varepsilon')^+$ follows. Similar, in each iteration of the Levenberg-Marquardt method the following minimization problem has to be solved

$$d_i = \arg\min_{d_i} \| \varepsilon' d_i - \varepsilon \|_2 + \delta_i \| d_i \|_2 = \left(\begin{bmatrix} \varepsilon' \\ \delta_i I \end{bmatrix} \right)^+ [\varepsilon^T\ 0\ \ldots\ 0]^T \tag{9}$$

Remark 1 *The linear quadratic minimization problems (8) and (9) can be solved using QR-factorization, which is computationally much more efficient than computing the pseudo-inverses of* ε' *and* $[\varepsilon'^T \delta_i I]^T$, *see [2].*

The following definitions will be needed in the sequel.

Definition 1
The *condition number* of matrix A is defined as $\text{cond}(A) = \frac{\max(\sigma(A))}{\min(\sigma(A))}$ where $\sigma(A)$ are the singular values of A. If $\text{cond}(A)$ is very large A (*e.g.* $> 10^3$), is said to be *ill-conditioned.*

Definition 2
An *ill-conditioned* minimization problem is a minimization problem where ε' is ill-conditioned.

Ill-conditioned minimization problems are often troublesome to minimize, and the iterative search (4) usually converges much slower than for better conditioned minimization problems. Neural network minimization problems are often very ill-conditioned (see, *e.g.*, [6]) and that motivates the research presented in the following two sections.

3. SEPARABLE MINIMIZATION PROBLEMS

There are two kinds of parameters in (1). The model is *separable* with respect to the parameters $\{c_k\}$, *i.e.*, given the parameters $\{a_k\}$ then $\{c_k\}$ can be calculated exactly by the least squares method, without iterative search. The

idea to make use of the separabillity feature is not new, a good overview is [5] and further references can be found in there.

However, it has, to the authors best knowledge, not been applied to neural nets before. The derivation here deviates slightly from the one in [5], and especially it will be shown that the separated problem is better conditioned than the original minimization problem. This means that the separated algorithm can be expected to converge with less iterations, which motivates the method. In [5] it is shown that the computational burden *per iteration* is of the same order for the separated and the non-separated algorithm. Hence, the use of the algorithm can only be motivated because the iterations become more efficient in the meaning that a lower number of iterations will be needed to minimize the criterion (3).

It is possible to combine the separation method with any of the minimization methods mentioned in previous section and this will be done on an example to vizualise the obtained improvement.

The main idea is simple, the estimate of the separable parameters is described by

$$c(a) = (g)^{+}y. \tag{10}$$

This expression can be substituted into the model (1). Since the parameters c have been eliminated this gives a new parameterization of the model with less parameters than the original problem. We will now look into the details.

Divide the parameter update and the derivative with respect to the two subsets of parameters, $d_i = [d_i^{cT} d_i^{aT}]^T$ and $\varepsilon' = [\varepsilon_c \ \varepsilon_a]$ where the derivative sign $'$ has been suppressed for the sake of clarity.

Introduce the projection P_c onto the row space of ε_c

$$P_c = \varepsilon_c(\varepsilon_c)^{+} = \varepsilon_c(\varepsilon_c^T\varepsilon_c)^{-1}\varepsilon_c^T$$

and the complementary projection $Q_c = I - P_c$ onto the kernel of ε_c^T. Since $P_c + Q_c = I$ the norm in (8) can be re-written in the following way

$$\begin{aligned} \| \varepsilon' d_i - \varepsilon \|_2 &= \| (P_c + Q_c)[\varepsilon_c \ \varepsilon_a]d_i - (P_c + Q_c)\varepsilon \|^2 = \\ &\| (Q_c\varepsilon_a d_i^a - Q_c\varepsilon) + (\varepsilon_c d_i^c + P_c\varepsilon_a d_i^a - P_c\varepsilon) \|_2 = \\ &\| Q_c\varepsilon_a d_i^a - Q_c\varepsilon \|_2 + \| \varepsilon_c d_i^c + P_c\varepsilon_a d_i^a - P_c\varepsilon \|_2 \end{aligned} \tag{11}$$

In the second step $Q_c\varepsilon_c = 0$ and $P_c\varepsilon_c = \varepsilon_c$ were used and the last equality follows from the fact that $P_cQ_c = 0$. The first term of (11) can be minimized independently of the second term and instead of minimizing the second term one can use (10) to obtain c. Hence, if Gauss-Newton method is applied to the separated problem the parameter update direction becomes

$$d_i^a = \arg\min_{d_i^a} \| Q_c\varepsilon_a d_i^a - Q_c\varepsilon \|_2 = (Q_c\varepsilon_a)^{+}Q_c\varepsilon. \tag{12}$$

If the Levenberg-Marquardt algorithm is preferred one has to modify this expression in analogy with (8) and (9).

Given an initial estimate of $a_k^{(0)}$ the following algorithm applies

Algorithm 1 *Separable Gauss-Newton minimization algorithm*

1. *Compute* $c(a)$ *with (10).*
2. *Compute* ε_a *and* ε_c *and form* Q_c.
3. *Compute* d_i^a *according to (12) and compute a candidate* $a^{(i)}$ *with a step length* $\mu_i = 1$.
4. *Compute* $c(a^{(i)})$ *with (10) and check that* $V_N(\widehat{\theta}^{(i+1)}) < V_N(\widehat{\theta}^{(i)})$. *If not, repeat from 2 with a smaller* μ_i.
5. *If* $a^{(i)}$ *has not converged, repeat from 2.*

To obtain a separable Levenberg-Marquardt algorithm the steps 3 and 4 have to be modified in analogy with (8) and (9).

The following theorem shows that the separated minimization problem becomes better conditioned than the original one.

Theorem 1 *The separated minimization problem is better, or at least as good conditioned as the non-separated minimization problem.*

Proof: It is to be shown that

$$\text{cond}(Q_c\varepsilon_a) \leq \text{cond}([\varepsilon_c \ \varepsilon_a]). \tag{13}$$

To do that we will make use of the following easily shown facts. Let A be an $n \times n$ matrix. A reduced matrix $\tilde{A}$ is obtained from A by deleting one row and one column from A. Then $\max(\sigma(A)) \geq \max(\sigma(\tilde{A}))$ and $\min(\sigma(A)) \leq \min(\sigma(\tilde{A}))$ where $\sigma(\cdot)$ are the singular values of the matrix.

1. From the definition (13) it then follows that $\text{cond}(\tilde{A}) \leq \text{cond}(A)$.
2. If A is invertible $\text{cond}(A) = \text{cond}(A^{-1})$.
3. If A is $n \times m$ and $n > m$ then $\text{cond}(A^TA) = \text{cond}(A)^2$.

Using these facts one has

$$\text{cond}([\varepsilon_c \ \varepsilon_a])^2 = \text{cond}\left(\begin{bmatrix} \varepsilon_c^T\varepsilon_c & \varepsilon_c^T\varepsilon_a \\ \varepsilon_a^T\varepsilon_c & \varepsilon_a^T\varepsilon_a \end{bmatrix}\right) =$$

$$\text{cond}\left(\begin{bmatrix} A & B \\ C & (\varepsilon_a^T\varepsilon_a - \varepsilon_a^T\varepsilon_c(\varepsilon_c^T\varepsilon_c)^{-1}\varepsilon_c^T\varepsilon_a)^{-1} \end{bmatrix}\right) \geq$$

$$\text{cond}(\varepsilon_a^T\varepsilon_a - \varepsilon_a^T\varepsilon_c(\varepsilon_c^T\varepsilon_c)^{-1}\varepsilon_c^T\varepsilon_a = \text{cond}(\varepsilon_a^TQ_cQ_c\varepsilon_a) = \text{cond}(Q_c\varepsilon_a)^2. \tag{14}$$

In the second step the matrix is inverted, A, B, and C are sub-matrices without interest in this context and they are removed in the third step giving the inequality. The claim (13) follows directly from (14). □

As mentioned in the beginning of the section it is shown in [5] that the computational burden of a single iteration is not changed due to using the separated Algorithm 1. Instead, Theorem 1 gives the benefit of the separation method: a better conditioned minimization problem.

3.1. Example

A data set of 300 samples is obtained in the following way. First 300 input data with $\dim\varphi = 5$ are generated, $\{\varphi(t)\}_{t=1}^{300}$, from a Gaussian distribution with unit variance. Then the output data, $\{y(t)\}_{t=1}^{300}$ are obtained with a one hidden-layer feed-forward neural net with 5 inputs and 1 output. The parameters a_k are chosen randomly from a Gaussian distribution with variance 2 and c_k from a Gaussian distribution with variance 6. The data are then used

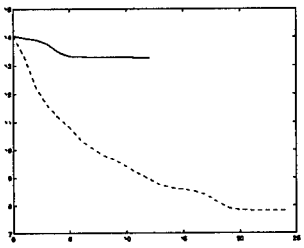


Figure 1: Criterion (3) as a function of the number of iterations of a Gauss-Newton search. Solid line: Standard Gauss-Newton method. Dashed line: Gauss-Newton applied to the separated problem.

to estimate a model of the same kind as the one which generated the data. The initial parameter values of the model are generated in the same way as described above. From the initial parameter values the criterion is iteratively minimized using the normal Gauss-Newton, and the separated Gauss-Newton method, Algorithm 1. The result is depicted in Figure 1. It is clear from the figure that the separated version succeeds much better. The standard Gauss-Newton terminates already after a few iterations. The reason for this is the ill-conditioning of the minimization problem and that will be explained in the following section.

4. CONVERGENCE SPEED OF MINIMIZATION ALGORITHMS

In this section an explanation will be given why the Levenberg-Marquardt algorithms often performs best on neural network minimization problems. Again it has to do with the ill-conditioning which has already been discussed.

It is well known that steepest descent search for the minimum is inefficient, especially for ill-conditioned problems close to the minimum. In such cases it is usually expected that the Newton, or Gauss-Newton method will perform better. See, *e.g.*, [1]. For these methods the matrix R_i in (5) transforms the search direction so that all parameters become equally important. This is illustrated in Figure 2, where the level curves of $V_N(\theta)$ are shown. Figure 2 a) corresponds to an ill-conditioned problem and the valley is a direction in the parameter space which has little influence on $V_N(\theta)$. The steepest descent method needs many iterations along the valley before the minimum is reached. The Gauss-Newton method on the other hand, transforms $V_N(\theta)$ so that all directions become equally important, depicted in Figure 2 b), and then the

convergence is typically much faster. Since the Gauss-Newton method rely on

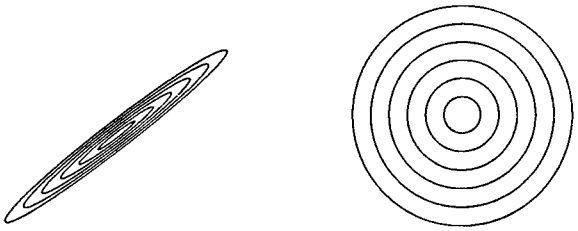


Figure 2: Level curves of the criterion $V_N(\theta)$, a) of an ill-conditioned problem, b) as seen by the Gauss-Newton method.

a quadratic Taylor expansion of $V_N(\theta)$ at $\widehat{\theta}^{(i)}$, the gain in convergence depends on how close $V_N(\widehat{\theta}^{(i)})$ is to be quadratic. It is clear that the quadratic Taylor expansion is good only in a neighborhood from the current estimate $\widehat{\theta}^{(i)}$, and a too large parameter step brings the estimate outside this region. Let us now look how the parameter steps look like with the different methods.

Since $R_i = \frac{1}{N}\varepsilon'^T\varepsilon'$ is symmetric it can be written $R_i = T_i Q_i T_i^T$, where the singular values, $\{q_k\}$, of R_i are the trace of the diagonal matrix Q_i and T_i consists of the eigenvectors. The parameter update direction (5) for the Levenberg-Marquardt method then becomes

$$d_i = (R_i + \delta I)^{-1}\nabla V_N(\widehat{\theta}^{(i)}) = T_i \mathrm{diag}\left(\frac{1}{q_1+\delta},\ldots,\frac{1}{q_d+\delta}\right) T_i^T \nabla V_N(\widehat{\theta}^{(i)}) \tag{15}$$

where d is the number of parameters. The corresponding update direction for the Gauss-Newton method is obtained by setting $\delta = 0$ in (15).

Given a $\delta > 0$, for those directions corresponding to small eigenvalues $\delta \gg q_k$ Gauss-Newton gives a large step size $1/q_k$. Levenberg-Marquardt on the other hand gives a small step of size $1/(q_k+\delta) \approx 1/\delta$. For those directions corresponding to large eigenvalues $\delta \ll q_k$ Levenberg-Marquardt gives a step size $1/(q_k+\delta) \approx 1/q_k$ approximately equal to the Gauss-Newton step. In this way Levenberg-Marquardt divides the parameter directions into two sub-classes. Within the first class one has an efficient convergence of "Gauss-Newton type", and within the second class one has a slow converging steepest descent method with step length $\mu_i = 1/\delta$. In this way one can consider the Levenberg-Marquardt method to be "in between" the Gauss-Newton method and the steepest descent method. What is the advantage of excluding some of the directions from the efficient fit?

There are two reasons to exclude the directions corresponding to the small singular values of Q_i from the parameter update step, *i.e.*, to exclude shallow valleys like in Figure 2 a) from the fit. First, in a direction with small q_k Gauss-Newton takes a large step $1/q_k$. This is likely to be a step outside the region where the second order Taylor expansion holds. Hence, to obtain a

decrease in the criterion $V_N(\theta)$ the step length μ_i in (4) must be small. The important parameters, corresponding to large q_k will then also be changed with the same small value μ_i instead with (for them) the optimal one ($\mu_i = 1$). This means that a very small step is taken in the direction where the criterion decreases at most, and with a small step size one has to perform a lot of iterations before the minimum is reached. It is clear that in such situations it is advantageous to divide the parameters in the way the Levenberg-Marquardt method does it.

The second reason to exclude those parameter directions which only influence the criterion $V_N(\theta)$ marginally has to do with the bias variance trade-off. Neural nets are often over-parameterized which means that they contain more parameters than necessary. The net does not benefit of the freedom which some of the parameters give it. Call these parameters ***spurious***, in contrast to the useful parameters. It is the spurious parameters which give the shallow valleys in the criterion function $V_N(\theta)$, and the spurious parameters will typically only fit the noise in the data. It is hence advantageous to exclude them from the fit. This can be done by using regularization or by stopping the minimization before the minimum is reached. See, *e.g.*, [7, 4].

It remains to decide upon the factor δ in (15). It is typically chosen by trail and error. Given an initial value of δ, if that value does not give a decrease of the criterion then it has to be increased. On the other hand, if it decreases the criterion, then a smaller value is tried in the following iteration.

Similar arguments can be used to explain why conjugate gradient minimization method, and even steepest descent method, can be better than Gauss-Newton method on ill-conditioned minimization problems.

4.1. Example

Let us continue with the example from the previous section. From the same initial parameter point the Levenberg-Marquardt and the steepest descent algorithms are applied – with the standard scheme and with the separated Algorithm 1. The results are depicted in Figure 3. Note that the y-axes are differently scaled. The results are strikingly in favor of the separated Levenberg-Marquardt algorithm.

5. CONCLUSIONS

The following conclusions can be done upon ill-conditioned neural net minimization problems:

- The conditioning of neural network minimization problems can be improved by separating the problem with respect to the linear parameters. This increases the convergence rate of the minimization.
- The Levenberg-Marquardt algorithm typically converges faster than the Gauss-Newton method due to that an efficient convergence is obtained

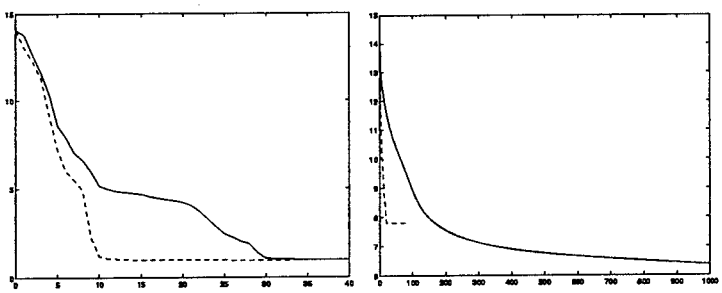


Figure 3: Criterion (3) as a function of the number of iterations of a) a Levenberg-Marquardt search b) Steepest-descent search. Solid line: Standard method. Dashed line: respectively method applied to the separated problem.

in the important parameter directions. At the same time the parameters which do not influence the criterion substantially are held almost fixed.

REFERENCES

[1] J.E. Dennis and R.B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Prentice-Hall, Englewood Cliffs, New Jersey, 1983.

[2] G. Golub and C. van Loan. *Matrix Computations.* The John Hopkins University Press, Baltimore, MD, USA, 1989.

[3] M.T. Hagan and M.B. Menhaj. Training feedforward networks with the Marquardt algorithm. *IEEE Trans. on Neural Networks*, 5(6):989–993, Nov. 1994.

[4] J.E. Moody. The effective number of parameters: An analysis of generalization and regularization in nonlinear learning systems. In J.E. Moody, S.J. Hanson, and R.P. Lippmann, editors, *Advances in Neural Information Processing Systems 4.* Morgan Kaufmann Publishers, San Mateo, CA, 1992.

[5] A. Ruhe and P.Å Wedin. Algorithms for separable nonlinear least squares problems. *SIAM Review*, 22(3):318–336, 1980.

[6] S. Saarinen, R. Bramley, and G. Cybenko. Ill-conditioning in neural network training problems. *SIAM Journal on Scientific Computing*, 14(3):693–714, May 1993.

[7] J. Sjöberg and L. Ljung. Overtraining, regularization, and searching for minimum with application to neural nets. *Int. J. Control*, 62(6):1391–1407, 1995.

[8] P.P. van der Smagt. Minimisation methods for training feedforward neural networks. *Neural Networks*, 7(1):1–11, 1994.

TRAINING RECURRENT NETWORKS

Morten With Pedersen
CONNECT, Department of Mathematical Modelling, Building 321
Technical University of Denmark, DK-2800 Lyngby, Denmark
Phone: + 45 45253920 Fax: + 45 45872599
email: mwp@imm.dtu.dk

Abstract - Training recurrent networks is generally believed to be a difficult task. Excessive training times and lack of convergence to an acceptable solution are frequently reported. In this paper we seek to explain the reason for this from a numerical point of view and show how to avoid problems when training. In particular we investigate ill-conditioning, the need for and effect of regularization and illustrate the superiority of second-order methods for training.

INTRODUCTION

Recurrent neural networks are an interesting class of models for signal processing as they are able to build up internal memory suited for the task at hand and thus often lead to compact model representations. However, it is generally believed to be a difficult task to train this type of networks. Several authors have addressed the learning problem for recurrent networks, e.g., in the context of sequence classification when required to store information for an arbitrary period of time [1, 5] but to the best of the authors knowledge no one have treated the problem from a general *numerical* point of view.

Feedforward networks were treated extensively from a numerical point of view in [7] where it was illustrated how training forms an extremely ill-conditioned optimization problem. In this contribution we extend this analysis to include recurrent networks. In particular we identify redundant connections and illustrate how ill-conditioning may otherwise arise, which motivates the use of regularization.

Having acknowledged the need for regularization makes way for the highly effective second-order methods for training. In this contribution we particulary focus on the *damped* Gauss-Newton method and illustrate how this method by far outperforms gradient descent on a time series prediction problem, namely the Santa Fe laser data. The focus in this contribution is on time series prediction, but the results generalize to other applications as well.

ARCHITECTURE

The general architecture of the networks considered here are fully connected feedback networks with one hidden layer of nonlinear units and a single linear

0-7803-4256-9/97/$10.00

output unit. The output $y(t)$ of the network is linear in order to allow for arbitrary dynamical range, and is given by

$$y(t) = \sum_{i=1}^{N_h} w_{oi} s_i(t) + w_{ob} \tag{1}$$

where N_h is the number of hidden units, w_{oi} is the weight to the output unit from hidden unit j and w_{ob} is a bias weight. The output $s_i(t)$ from hidden unit i at time t is computed as

$$s_i(t) = f\left(\sum_{j=1}^{N_h} w_{ij} s_j(t-1) + w_{io} y(t-1) + \sum_{k=1}^{N_I} w_{ik} x_k(t) + w_{ib}\right) \tag{2}$$

where w_{ij} is the weight *to* hidden unit i *from* hidden unit j, w_{io} is the weight *to* hidden unit i *from* the output unit and w_{ib} is the bias weight for hidden unit i. $x_k(t)$ is the k'th element in the external input vector $\mathbf{x}(t)$ at time t and N_I is the total number of external inputs. $f(\cdot)$ is the nonlinear activation function, in this work we use $f(x) = \tanh(x)$.

Note that the update of the recurrent network presented above is *layered*, as the outputs $s_i(t)$ from the hidden units are computed immediately *before* the computation of the output unit output. This is opposed to the update presented in e.g. [10] where all the units are updated simultaneously. In [6] it was shown that when using fully recurrent networks for forecasting, layered update is preferable since synchronous update of the units effectively results in a two-step ahead predictor. Note also that the linear output unit does not have feedback of its own previous value. This is in order to avoid stability problems that are otherwise likely to occur.

Training

In this work we focus on time series prediction in which case the input vector contains delayed elements of the time series, $\mathbf{x}(t) = [x(t), \ldots, x(t - N_I + 1)]$, and the network output is a prediction of the next value in the series, $\widehat{x}(t+1) = y(t)$. Training the network means adjusting the weights so as to minimize a cost function. Most applications are based on the sum of squared errors,

$$E(\mathbf{w}) = \frac{1}{2}\sum_{t=1}^{T}[e(t)]^2 \ , \ e(t) = x(t+1) - y(t) \tag{3}$$

where T denotes the number of training examples and $\mathbf{w}$ is the concatenated set of parameters. The adjustment of the parameters is done *off line* by an iterative sheme, $\mathbf{w}_{k+1} = \mathbf{w}_k - \eta \Delta \mathbf{w}_k$, where $\Delta \mathbf{w}_k$ indicates the direction of change and η is the (adaptive) size of the step. When training recurrent neural networks the most commonly used scheme is gradient descent, where the direction $\Delta \mathbf{w}_k$ is equal to the gradient $\mathbf{g}$, $g_i = \partial E(\mathbf{w}_k)/\partial w_i$. Unfortunately this method suffers from extremely slow convergence, and the quality of resulting solutions is often not satisfactory.

Experiments have shown that much more efficient training can be obtained by using second-order methods [6]. Here we focus on the *damped* Gauss-Newton method [3], in which the search direction $\Delta \mathbf{w}_k$ is determined by

$$\Delta \mathbf{w}_k = \mathbf{H}^{-1}\mathbf{g} \tag{4}$$

where $\mathbf{H}$ is the positive semidefinite approximation to the Hessian,

$$H_{ij} = \sum_{t=1}^{T} \frac{\partial y(t)}{\partial w_i}\frac{\partial y(t)}{\partial w_j} \approx \frac{\partial^2 E(\mathbf{w}_k)}{\partial w_i \partial w_j} = \sum_{t=1}^{T}\left[\frac{\partial y(t)}{\partial w_i}\frac{\partial y(t)}{\partial w_j} - e(t)\frac{\partial^2 y(t)}{\partial w_i \partial w_j}\right] \tag{5}$$

In each iteration k the step size η is determined by line search which makes the method globally convergent [3]; here we recommend a simple approach where η is halved until a decrease in the cost is obtained [3]. The iterations are continued until convergence, determined by a sufficiently small length of the gradient, $\|\mathbf{g}\|_2 < \varepsilon$. The Gauss-Newton method involves finding the solution to a linear system of equations $\mathbf{H}\Delta \mathbf{w}_k = \mathbf{g}$ in each iteration, but the increased computational burden is justified by a dramatic increase in convergence and thus reduction of overall training time, even for large networks as we shall see. However, the success of the damped Gauss-Newton method relies heavily on the conditioning of the training problem, as is the case for gradient descent.

ILL-CONDITIONING

When training using either gradient descent or the Gauss-Newton method, a measure of great importance for the convergence is the condition number of the Hessian $\mathbf{H}$. For a symmetric positive definite matrix $\mathbf{H}$, the condition number is defined as $\kappa(\mathbf{H}) = \lambda_{max}/\lambda_{min}$, the ratio between the largest and smallest eigenvalue of $\mathbf{H}$. If the condition number is *large*, the Hessian becomes *ill-conditioned.* The convergence rate will suffer and the solution to the linear system of equations (4) in the Gauss-Newton method becomes unreliable. As a rule of thumb the solution may not be trustworthy if $\kappa(\mathbf{H}) > \epsilon^{-1/2}$, where ϵ denotes the machine precision [3]. For the IEEE 64-bit floating point representation this is equivalent to $\kappa(\mathbf{H}) > 6.7 \cdot 10^7$. This may seem as a large number, but this order of magnitude is not uncommon in the framework of either feedforward networks [7] or recurrent networks as we shall see.

In [2] it was shown that an eigenvalue of the order of the number of input variables could be avoided if the mean was subtracted from each of the input variables $x_k(t)$ and if a symmetric activation function is used. However, these simple countermeasures are not adequate for avoiding ill-conditioning in recurrent networks, as the analysis in the following will show.

The Hessian (5) can also be written as

$$\mathbf{H} = \mathbf{J}^{\mathrm{T}}\mathbf{J}\ ,\ J_{ti} = \frac{\partial y(t)}{\partial w_i} \tag{6}$$

where $\mathbf{J}$ is the Jacobian matrix, whose columns are the partial derivatives of the network output at each timestep in the training series. If $\mathbf{J}$ is rank-deficient some of the columns are linearly dependent, which is indicated by

singular values with the value zero in an SVD analysis. This again leads to a singular Hessian and thus an infinite condition number. In practice it is rare to find columns in $\mathbf{J}$ that are *exactly* dependent and thus singular values that are exactly zero [7]. However, it is often the case that columns are *nearly* linearly dependent, which leads to very small singular values of $\mathbf{J}$ and thus large condition numbers for the Hessian $\mathbf{H}$. In the following sections we describe situations leading to ill-conditioning of $\mathbf{J}$ for recurrent networks, arising from both exact and approximate linearly dependencies between columns in $\mathbf{J}$.

Exact dependency

For the type of recurrent networks defined by (1) and (2) there is built-in rank deficiency in the Jacobian since it is easy to show that some of the columns in $\mathbf{J}$ will *always* be linear combinations of each other. This is illustrated by an example for a small network, but the result apply for networks with an arbitrary number of hidden units. The network considered here involves only one external input and one hidden unit, and the output is thus defined as

$$y(t) = w_{o1}s_1(t) + w_{ob} \tag{7}$$

$$s_1(t) = f\left(w_{11}s_1(t-1) + w_{1o}y(t-1) + w_{1x}x(t) + w_{1b}\right) \tag{8}$$

$$= f\left((w_{11} + w_{1o}w_{o1})s_1(t-1) + w_{1x}x(t) + (w_{1b} + w_{1o}w_{ob})\right) \tag{9}$$

where (9) is obtained by insertion of (7) in (8). We see that the network output will remain unchanged as long as the total weighting k_1 of $s_1(t-1)$, $k_1 = w_{11} + w_{1o}w_{o1}$, and the total bias k_2 on the hidden unit, $k_2 = w_{1b} + w_{1o}w_{ob}$, remains constant. w_{o1} and w_{ob} can not be changed without directly affecting the network output (7) and are therefore kept fixed which we denote by $*$. However, changes in w_{11}, w_{1o} and w_{1b} that satisfies *both* expressions

$$\begin{array}{ccccccc} w_{11} & + & w^*_{o1}\cdot w_{1o} & + & 0 & = & k_1 \\ 0 & + & w^*_{ob}\cdot w_{1o} & + & w_{1b} & = & k_2 \end{array} \tag{10}$$

will leave the network output unchanged. The expressions (10) form hyperplanes in parameter space spanned by w_{11}, w_{1o} and w_{1b} and their line of intersection is computed as $(w_{11}, w_{1o}, w_{1b}) = (k_1, 0, k_2) + t(-w^*_{o1}, 1, -w^*_{ob})$, parametrized by t. The line defines a direction in parameter space in which the network output is constant. The constant network output means that derivatives are zero in this direction. Thus, columns in the Jacobian corresponding to (w_{11}, w_{1o}, w_{1b}) are linearly dependent.

When investigating Jacobians for the dependency problem outlined above it is however uncommon to encounter singular values *exactly* equal to zero; but according to the derivations this clearly ought to be the case. The reason for this is the initialization of previous state values when starting up the network. If the recurrent network starts iteration at time $t = 1$ it is common practice [10] to set the previous states of the hidden units as well as their derivatives to zero, $s_i(0) = 0$, $\partial s_i(0)/\partial \mathbf{w} = 0$. This startup procedure clearly marks an initial discontinuity in the recursive equations (7) and (8)

governing the feedback network. Thus initially the partial derivatives wrt. the involved weights in the Jacobian will generally *not* be linearly dependent. But after a few iterations indicating a transient, the dependency arises with increasing accuracy. The linear dependency is eliminated if we omit the feedback weights w_{io} *from* the output *to* the hidden units i, as the degeneracy can then no longer occur. This elimination has no influence on the modeling capabilities of the network since the remaining weights can be adjusted so that the network output remains unaffected.

Approximate dependency

Even though removal of the feedback weights w_{io} leading from the linear output to the hidden units removes the problem of almost exact rank-deficiency in the Jacobian for recurrent networks it does not eliminate ill-conditioning as experiments show. In [7] the problem of ill-conditioning was analyzed for *feedforward* networks by careful examination of the components entering the partial derivatives $\partial y(t)/\partial w_i$ of the network output and it was found that ill-conditioning in the Jacobian can arise from at least these three reasons (assuming that the external inputs are not proportional):

1. The output from a hidden unit is saturated and constant ($\equiv \pm 1$).
2. The outputs from two hidden units are approximately proportional.
3. The derivatives of two hidden unit outputs wrt. their activations are approximately proportional.

Theoretical and empirical examinations of the components entering the partial derivatives for *recurrent* networks reveal that ill-conditioning may arise here from the same reasons; such analysis is however not included here.

Situation 2 where the outputs of two hidden units are proportional and thus highly correlated often occurs in practice; e.g., in [9] high correlation between hidden unit outputs was found and studied for feedforward networks.

The effects of situation 2 are similar to the effects of exact dependencies described above, as we can determine directions in parameter space in which the cost function is approximately constant. For recurrent networks this situation is much more severe than for feedforward networks since the degeneracy will not only affect weights leading to the output, but also many weights connecting the hidden units as the experiments will show.

The scenarios listed above lead to nearly linearly dependencies between the columns of $\mathbf{J}$ and thus to small eigenvalues in $\mathbf{H}$. However, the condition number of a matrix is determined by the *ratio* between the largest and smallest eigenvalues, thus problems do not only arise from small singular values but also from large values. As mentioned, the situations described above will lead to directions in parameter space where the cost is approximately constant, thus when training using the Gauss-Newton method the search direction will be dominated by these directions leading to an unrestrained growth in the magnitude of the affected weights. This again leads to a significant growth in

the magnitude of several of the columns in the Jacobian since many derivatives are dominated by terms of the form [10]

$$\frac{\partial s_k(t)}{\partial w_{pq}} \propto \sum_{j=1}^{N_h} w_{kj} \frac{\partial s_j(t-1)}{\partial w_{pq}} \tag{11}$$

which becomes large if the weights w_{kj} become large. The large elements in the Jacobian lead to an overall upward scale of the elements in $\mathbf{J}^{\mathrm{T}}\mathbf{J}$ and thus to an upward shift of the eigenvalues.

REGULARIZATION

A traditional method for handling ill-conditioning is by *regularizing* the cost function [3, 4]. A simple yet highly effective regularization can be obtained by augmenting the cost function by a simple quadratic weight decay [4],

$$C(\mathbf{w}) = E(\mathbf{w}) + \frac{\alpha}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w} \tag{12}$$

Simple weight decay is often primarily considered as a means for avoiding overfitting as it puts constraints on the parameters and thus reduces the degrees of freedom. Weight decay should however also be considered from its regularizing effects. The immediate effect is that α gets added to the diagonal of the Hessian which puts a lower bound on the smallest eigenvalues, since it is easy to show that $\lambda(\mathbf{H} + \alpha\mathbf{1}) = \lambda(\mathbf{H}) + \alpha$. Another effect is the limit imposed on the growth of the weights which prevents near singular directions in parameter space from dominating the search directions obtained by the Gauss-Newton method, thus greatly improving the efficiency of the optimization. The constraints put on the weights by the regularization has a smoothing effect on the cost function which was clearly illustrated in [6]. Here it was also demonstrated that the significance of the second order term ignored in (5) diminishes when using simple weight decay as regularization.

EXPERIMENTS

In the first experiment we illustrate how ill-conditioning results from some of the situations described herein and how regularization improves training. For this experiment we used a simple recurrent network to predict the laser data from the Santa Fe time series prediction competition [8]. The data were scaled so that the first 1000 points used for training had zero mean and unit variance and the following 100 values were used as a test set. The network used had one external input and three hidden units; there were *no feedback from the linear output unit* to the hidden units as found appropriate above. Training was performed initially using five iterations of gradient descent followed by the damped Gauss-Newton method. In the left panel of Figure 1 is shown the evolution of the *mean* squared errors normalized by the variance of the sets (NMSE, [8]) when training without regularization. It seems that training is converging to a solution, but this is *not* the case as the evolution of the weights

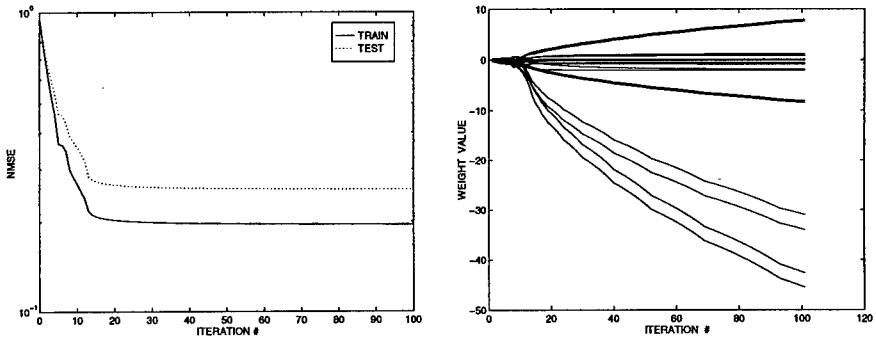


Figure 1: Training without regularization. Left panel: Evolution of training and test errors. Right panel: Evolution of the weight values.

in the right panel of Figure 1 shows. What happens is that the outputs of two hidden units become almost proportional; this is revealed by the cosine to the angle θ between vectors containing their outputs on the training set which at iteration 100 is $\cos\theta = 0.9998$. This corresponds to situation 2 listed above. The weights that grow in magnitude are the pairs of weights leading *from* these two units *to* every unit in the network including the output. Note that the error and thus the network output is unaffected since the effects of the changes in the growing weights cancel out due to the dependency between the hidden units.

The condition number during training is shown in the left panel of Figure 2 and is seen to grow enormously. The rapid increase occurs shortly after the initiation of the second-order method which quickly 'discovers' the dependency between the hidden unit outputs. The near singular Hessian $\mathbf{H}$ leads to very large weight changes in some directions when solving (4). The large steps are however handled by the line search which returns very small step sizes, indicated by the smooth increase in the weight magnitudes. In the right panel of Figure 2 is shown the eigenvalues of the Hessian after iterations 7, 20 and 100. At each of the iterations it is seen that the condition number results from both very small as well as very large eigenvalues and we note that as

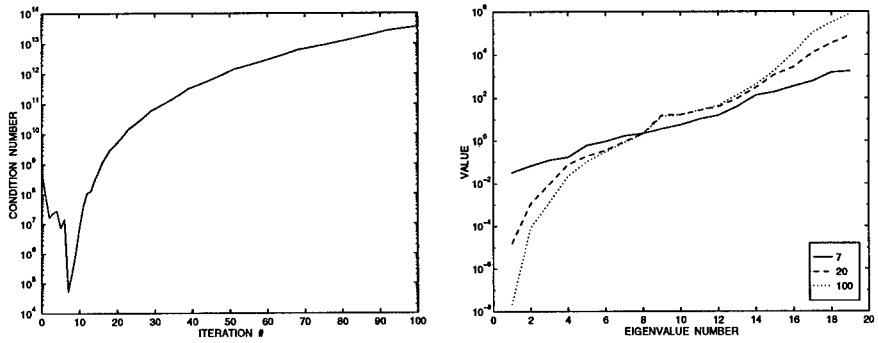


Figure 2: Training without regularization. Left panel: Evolution of the condition number for $\mathbf{H}$. Right panel: Eigenvalues after iterations 7, 20 and 100.

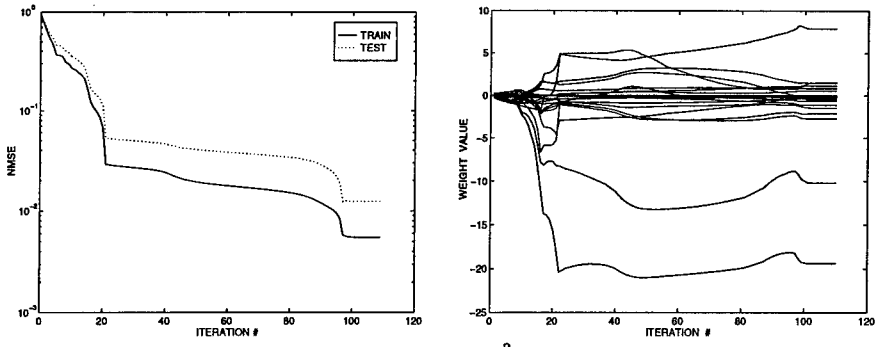


Figure 3: Training with regularization, $\alpha = 10^{-3}$. Left panel: Evolution of training and test errors. Right panel: Evolution of the weight values.

training progresses the eigenvalues extend both upward and downward.

The training was then repeated using the exact same initial weights and the same training approach, but now with a regularization term added to the cost function, using $\alpha = 10^{-3}$. In the left panel of Figure 3 is shown the resulting evolution of the errors. The positive effect of the regularization is evident, as the final errors are several orders of magnitude below the levels shown in Figure 1 obtained without regularization. Furthermore the stopping criterion $\|\mathbf{g}\|_2 < 10^{-4}$ was satisfied; in the previous experiment using no regularization the gradient norm grew proportional to the condition number.

In the right panel of Figure 3 we see that the regularization term limits the growth of the weights compared to Figure 1. Some however still grow large as does the condition number shown in the left panel of Figure 4. Even though the condition number grows to 10^8 the damped Gauss-Newton method still manages to find a minimum. Experience shows that for this method successful training to a (local) minimum can be obtained for condition numbers up to about 10^8 in magnitude. This may depend on the decomposition algorithm used when solving (4), here we use the fast and stable Cholesky factorization [3]. From the right panel of Figure 4 we learn that the reduction in condition number is obtained *only* from an increase in the smallest eigenvalues resulting

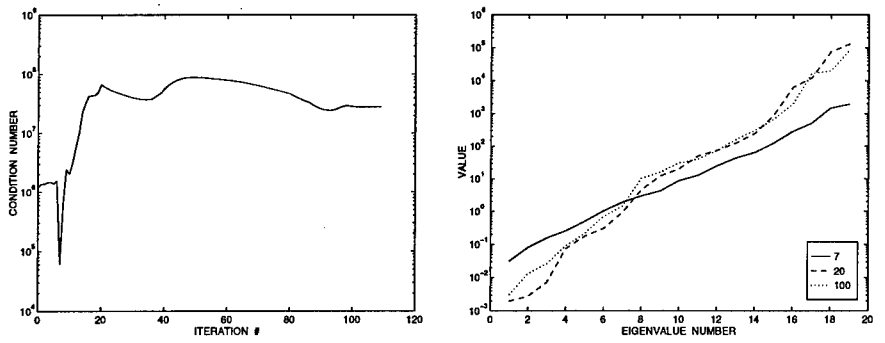


Figure 4: Training with regularization, $\alpha = 10^{-3}$. Left panel: Evolution of the condition number for $\mathbf{H}$. Right panel: Eigenvalues after iterations 7, 20 and 100.

from the regularization. The largest eigenvalues are of the same order of magnitude as when training without weight decay, see Figure 2. This is due to the still fairly large weight magnitudes. If the regularization term α is further increased the larger eigenvalues will also be affected; but so will the modeling capabilities of the network, leading to increased errors.

In the final experiment we compare the performance of damped Gauss-Newton with a gradient descent algorithm also using the step-size halving line search. The problem is still prediction of the laser series but using larger networks with a single input and nine hidden units, 109 weights in total (no feedback from the output to the hidden units). Thus, each iteration using damped Gauss-Newton involved solution of a 109 by 109 linear system of equations. Six initial networks were generated by initializing their weights with values drawn from a uniform distribution over the interval $[-0.3; 0.3]$. The training algorithms were then compared when starting from the same six initial networks, both using regularization $\alpha = 0.02$. The resulting evolution of errors is shown in Figure 5; in the left panel we see the resulting errors using the damped Gauss-Newton method, in the right panel µsing gradient descent. Using both methods the stopping criteria was set to $\|\mathbf{g}\|_2 < 10^{-4}$ or maximum 10000 iterations.

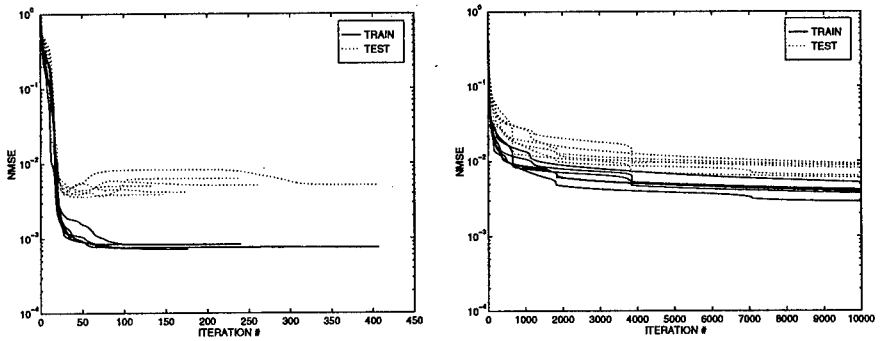


Figure 5: Evolution of errors using different optimization methods. Left panel: Damped Gauss-Newton method. Right panel: Gradient descent with line search.

For the damped Gauss-Newton method the stopping criterion was met in all six runs. The average training error (Normalized Mean Squared Error) was $7.7 \cdot 10^{-4}$, the average test error was $4.9 \cdot 10^{-3}$. The *average* time for a complete training run was 200 seconds. For gradient descent the stopping criterion was never met, the termination of the algorithm in each run was due to maximum number of iterations reached. The average training error obtained after the maximum allowed 10000 iterations was $4.0 \cdot 10^{-3}$, the average test error was $7.8 \cdot 10^{-3}$. The average time used for obtaining these error levels was 8140 seconds. Note that the levels of both training and test errors obtained using gradient descent are much higher than the levels obtained using the damped Gauss-Newton method even though gradient descent used a factor of 50 times more iterations and a factor of 40 times more computer time. Thus, even though an iteration of the damped Gauss-Newton method is computationally

more costly than an iteration of gradient descent, the additional cost is highly justified by the vastly increased convergence rate. Similar justification has been observed for networks with up to 300 parameters.

CONCLUSION

In this paper we have focused on sources of ill-conditioning and thus the need for regularization when training recurrent networks especially using second-order methods. Once this need is recognized dramatic improvement in convergence rate and quality of solution is obtained, even for large size problems.

ACKNOWLEDGMENTS

The author would like to thank Lars Kai Hansen and Jan Larsen for support. This research is supported by the Danish Natural and Technical Research Coucils through the Computational Neural Network Center (CONNECT).

REFERENCES

[1] Y. Bengio, P. Simard and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," **IEEE Transactions on Neural Networks**, vol. 5, no. 2, pp. 157–166, 1994.

[2] Y. L. Cun, I. Kanter and S. A. Solla, "Eigenvalues of covariance matrices: Application to neural-network learning," **Physical Review Letters**, vol. 66, no. 18, pp. 2396–2399, 1991.

[3] J. E. Dennis and R. B. Schnabel, **Numerical Methods for Unconstrained Optimization and Nonlinear Equations**, Englewood Cliffs, NJ: Prentice-Hall, 1983.

[4] S. Haykin, **Neural Networks, A Comprehensive Foundation**, New York, NY: Macmillan, 1994.

[5] S. Hochreiter and J. Schmidhuber, "Long short term memory," Tech. Rep. FKI-207-95, Fakultat fur Informatik, Technische Universitat Munchen, Munchen, 1995.

[6] M. W. Pedersen and L. K. Hansen, "Recurrent networks: Second order properties and pruning," in G. Tesauro, D. Touretzky and T. Leen, eds., **Advances in Neural Information Processing Systems**, The MIT Press, 1995, vol. 7, pp. 673–680.

[7] S. Saarinen, R. Bramley and G. Cybenko, "Ill-conditioning in neural network training problems," **SIAM Journal on Scientific Computing**, vol. 14, pp. 693–714, 1993.

[8] A. S. Weigend and N. A. Gershenfeld, eds., **Time Series Prediction: Forecasting the Future and Understanding the Past**, Reading, MA: Addison-Wesley, 1993.

[9] A. S. Weigend and D. E. Rumelhart, "The effective dimension of the space of hidden units," in E. Keramides, ed., **Proceedings of INTERFACE'91: Computing Science and Statistics**, Springer Verlag, 1992.

[10] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," **Neural Computation**, vol. 1, pp. 270–280, 1989.

COMBINING DISCRIMINANT-BASED CLASSIFIERS USING THE MINIMUM CLASSIFICATION ERROR DISCRIMINANT

Naonori UEDA Ryohei NAKANO

NTT Communication Science Laboratories
Hikaridai Seika-cho Soraku-gun Kyoto 619-02 Japan
e-mail: ueda@cslab.kecl.ntt.jp

Abstract. Focusing on classification problems, this paper presents a new method for linearly combining discriminant-based classifiers to the improve classification performance, in the sense of the minimum classification errors. In our approach, the problem of estimating linear weights in combination is reformulated as the problem of designing a linear discriminant function using the minimum classification error discriminant. In this formulation, because the classification decision rule is incorporated into the cost function, better combination weights suitable for the classification objective can be obtained. Experimental results using neural network classifiers support the effectiveness of the proposed method.

INTRODUCTION

Compared with a single estimator, combining estimators has been shown to better improve generalization error [1]-[8]. Approaches to combine estimators have recently attracted major interest in the neural network community because of thier simplicity and theoretical implications. The output of the combined estimator for some input is defined as a linear combination of outputs of multiple estimators, where it is assumed that each estimator is separately constructed by using the same training data.

In these approaches, how to determine the linear weights is an important problem in practice. A naive way is to employ simple averaging (*i.e.*, equal weights). Recently, a few methods have been presented [4][8] to combining regressors. However, as we will describe later, these methods are not suitable for our problem of combining classifiers.

In this paper, we newly present a way of combining discriminant-based classifiers to improve the classification performance. In our approach, the problem of estimating linear weights in combination is reformulated as the problem of designing linear discriminant functions using the minimum classification error discriminant. Since the decision rule is directly incorporated into

0-7803-4256-9/97/$10.00

the cost function in this formulation, we can obtain linear weights with which a better combined classifier can be constructed for achieving the minimum misclassification probability.

This minimum classification error discriminant approach has successfully been utilized to train individual discriminant functions [9][10]. We newly present how this criterion can be used to estimate the linear weights in combining discriminant-based classifiers.

PROBLEM FORMULATION

Let $\boldsymbol{x}$ be an observation vector, and let it be our purpose to assign $\boldsymbol{x}$ to one of K classes. A decision rule in terms of discriminant functions is written as follows:

$$C(\boldsymbol{x}) = k \quad \text{if} \quad f^{(k)}(\boldsymbol{x}) = \max_j f^{(j)}(\boldsymbol{x}). \tag{1}$$

Here $C(\cdot)$ denotes a classification operation and $f^{(k)}(\boldsymbol{x})$ is the discriminant function for class k. Thus, a classifier consists of K discriminant functions. The simplest instance is the linear discriminant function specified by weight vector $\boldsymbol{w}$ as $f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x}$, where T denotes the transposition. Neural network classifiers might be the most complicated. In the case of a neural network classifier, the kth output unit corresponds to the discriminant function for class k.

Suppose we have M available classifiers, which were trained using the *same* training data $\mathcal{D} = \{(\boldsymbol{x}_i, C(\boldsymbol{x}_i));\ i = 1, \ldots, N\}$. Let $f_m^{(k)}(\boldsymbol{x}; \mathcal{D})$ denote the output of the mth discriminant function for class k for some input $\boldsymbol{x}$ after it has been trained on $\mathcal{D}$. Then, in an analogous manner to the original definition of the linear combination of multiple regressors [2], the combined discriminant function for each class can be defined as the linear combination of all KM discriminant functions:

$$f_{com}^{(k)}(\boldsymbol{x}) \stackrel{\text{def}}{=} \boldsymbol{\alpha}^{(k)T}\boldsymbol{f}(\boldsymbol{x}; \mathcal{D}), \quad k = 1, \ldots, K. \tag{2}$$

Here,

$$\boldsymbol{f}(\boldsymbol{x}; \mathcal{D}) = \left(f_1^{(1)}(\boldsymbol{x}; \mathcal{D}), \ldots, f_1^{(K)}(\boldsymbol{x}; \mathcal{D}), \ldots, f_M^{(1)}(\boldsymbol{x}; \mathcal{D}), \ldots, f_M^{(K)}(\boldsymbol{x}; \mathcal{D})\right)^T \tag{3}$$

and therefore $\boldsymbol{\alpha}^{(k)}$ is a KM-dimensional column vector. Another definition such as $f_{com}^{(k)} = \sum_{m=1}^{M} \alpha_m^{(k)} f_m^{(k)}(\boldsymbol{x}; \mathcal{D})$, where $k = 1, \ldots, K$ is possible, but our definition is more flexible and general.

Let $\boldsymbol{y}(\boldsymbol{x}) = \left(f_{com}^{(1)}(\boldsymbol{x}), \ldots, f_{com}^{(K)}\right)^T$ be a classifier vector. Then, (2) can be written as

$$\boldsymbol{y}(\boldsymbol{x}) = W^T\boldsymbol{f}(\boldsymbol{x}; \mathcal{D}), \tag{4}$$

where $W = (\boldsymbol{\alpha}^{(1)}, \ldots, \boldsymbol{\alpha}^{(K)})$. Therefore, one can see that (4) indicates a linear mapping from a $f \in \mathcal{R}^{KM}$-space to a $\boldsymbol{y} \in \mathcal{R}^K$-space.

Now our goal here is to estimate $\boldsymbol{\alpha}^{(k)}$, where $k = 1, \ldots, K$ to design a combined classifier achieving the minimum classification error probability.

PREVIOUS WORK

Variance-based Weighting

Theoretically, if the prediction values of individual estimators are uncorrelated and unbiased, the combined prediction value becomes unbiased. Moreover, the smallest generalization error at $\boldsymbol{x}$ can be attained only if the weighting function is selected as the inverse of the variance of the prediction values at $\boldsymbol{x}$ [2][4][7]. Tresp *et al.* [4] proposed a non-constant weighting function defined as the inverse of the variance depending on $\boldsymbol{x}$. This method is promising in problems involving the usual combining of regressors denoted as $g_{com}(\boldsymbol{x}) = \sum_{m=1}^{M} g_m(\boldsymbol{x}; \mathcal{D})$. In the case of our definiton for combination (2), however, for all k the same linear weights:

$$\boldsymbol{\alpha}^{(k)}(\boldsymbol{x}) \propto \left(1/\mathrm{var}(f_1^{(1)}(\boldsymbol{x}; \mathcal{D})), \ldots, 1/\mathrm{var}(f_M^{(K)}(\boldsymbol{x}; \mathcal{D}))\right)^T$$

are assigned and as a result $f_{com}^{(1)}(\boldsymbol{x}) = f_{com}^{(2)}(\boldsymbol{x}) = \cdots = f_{com}^{(K)}(\boldsymbol{x})$. This is clearly inappropriate. This might be modified by considering the *bias* of $f_m^{(k)}(\boldsymbol{x}; \mathcal{D})$, but generally it is much more difficult to estimate the bias. Consequently, the variance-based weighting method is not directly applicable to our task.

Stacked Regression (MSE-criterion)

Let $\boldsymbol{b}(\boldsymbol{x})$ be a K dimensional column vector whose kth component is one; and the others are zero if $C(\boldsymbol{x}) = k$. Let X be a random variable having the distribution $p(\boldsymbol{x})$. In a regression setting, the mean squared error (MSE) of combined classifier f_{com} is written as

$$\mathrm{MSE}(f_{com}) = \mathrm{E}_X \left\{ \left\| \boldsymbol{b}(X) - \left(\boldsymbol{\alpha}^{(1)^T} \boldsymbol{f}(X; \mathcal{D}), \ \ldots, \ \boldsymbol{\alpha}^{(K)^T} \boldsymbol{f}(X; \mathcal{D}) \right)^T \right\|^2 \right\}. \tag{5}$$

Therefore, one way for estimating the weights with $\mathcal{D}$ is just to take $\boldsymbol{\alpha}^{(k)}$, where $k = 1, \ldots, K$ to minimize

$$\frac{1}{N} \sum_{i=1}^{N} \left\| \boldsymbol{b}(\boldsymbol{x}_i) - \left(\boldsymbol{\alpha}^{(1)^T} \boldsymbol{f}(\boldsymbol{x}_i; \mathcal{D}), \ \ldots, \ \boldsymbol{\alpha}^{(K)^T} \boldsymbol{f}(\boldsymbol{x}_i; \mathcal{D}) \right)^T \right\|^2. \tag{6}$$

However, as pointed out by Breiman [5], since $\mathcal{D}$ is utilized both in the training of each $f_m^{(k)}$ and in the estimation of $\boldsymbol{\alpha}$, the obtained $f_{com}^{(k)}$ will overfit the training data $\mathcal{D}$. This will result in poor generalization. He also presents a method called *stacked regression* to fix this problem, based on the idea of *stacked generalization* by Wolpert [1].

Let $f_m^{(k)}(\boldsymbol{x}_i; \mathcal{D}_{(-i)})$ denote the cross-validated output evaluated at the left-out a data $\boldsymbol{x}_i$ after $f_m^{(k)}$ has been trained on $\mathcal{D}_{(-i)} \equiv \mathcal{D} - \{(\boldsymbol{x}_i, C(\boldsymbol{x}_i))\}$. The stacked regression estimates of the weights can be obtained by minimizing

$$\frac{1}{N}\sum_{i=1}^{N}\left\|\boldsymbol{b}(\boldsymbol{x}_i) - \left(\boldsymbol{\alpha}^{(1)^T}\boldsymbol{f}(\boldsymbol{x}_i; \mathcal{D}_{(-i)}),\ \ldots,\ \boldsymbol{\alpha}^{(K)^T}\boldsymbol{f}(\boldsymbol{x}_i; \mathcal{D}_{(-i)})\right)^T\right\|^2. \quad (7)$$

The above MSE solution can be easily obtained as

$$\hat{W}_{MSE} = \left(\sum_{i=1}^{N}\boldsymbol{f}_i\boldsymbol{f}_i^T\right)^{-1}\left(\sum_{i=1}^{N}\boldsymbol{f}_i\boldsymbol{b}_i^T\right), \quad (8)$$

where

$$\begin{aligned}\boldsymbol{f}_i = \boldsymbol{f}(\boldsymbol{x}_i; \mathcal{D}_{(-i)}) \ \equiv\ & (f_1^{(1)}(\boldsymbol{x}_i; \mathcal{D}_{(-i)}), \ldots, f_1^{(K)}(\boldsymbol{x}_i; \mathcal{D}_{(-i)}), \\ & \ldots, f_M^{(1)}(\boldsymbol{x}_i; \mathcal{D}_{(-i)}), \ldots, f_M^{(K)}(\boldsymbol{x}_i; \mathcal{D}_{(-i)}))^T. \quad (9)\end{aligned}$$

Looking at (2) carefully, one can see that $f_{com}^{(k)}$ is a linear discriminant function in an $\boldsymbol{f}$-space. According to the theory of statistical pattern recognition [11], *i.e.*, canonical discriminant analysis, the MSE criterion produces the same solution as Fisher's linear discriminant criterion. In other words, a combined classifier based on the MSE approach is exactly equivalent to Fisher's linear discriminant function. It is known that Fisher's criterion works well as a *class separability measure* under the assumption that the distribution of each class is normal. In our problem, however, this assumption does not always hold and therefore it is clear that the MSE criterion does not always lead to the optimal solution in the Bayes sense.

ESTIMATING WEIGHTS USING THE MINIMUM CLASSIFICATION ERROR DISCRIMINANT

Formulation

As mentioned in the previous section, since $f_{com}^{(k)}$ can be seen as a *linear* discriminant function parameterized by $\boldsymbol{\alpha}^{(k)}$ in an $\boldsymbol{f}$-space, the problem of estimating the linear weights is exactly equivalent to the problem of designing a linear classifier in an $\boldsymbol{f}$-space. Thus our problem is reduced to designing K linear discriminant functions $\boldsymbol{\alpha}^{(1)^T}\boldsymbol{f}$, $\ldots$, $\boldsymbol{\alpha}^{(K)^T}\boldsymbol{f}$ from cross-validated data $D' = \{(\boldsymbol{f}(\boldsymbol{x}_i; \mathcal{D}_{(-i)}), C(\boldsymbol{x}_i));\ i = 1, \ldots, N\}$. Note that once D' is constructed, we do not need D any more to estimate W. To emphasize this, we write $D' = \{(\boldsymbol{f}_i, C(\boldsymbol{f}_i)) \in \mathcal{R}^{KM+1};\ i = 1, \ldots, N\}$ by setting $\boldsymbol{f}_i \equiv \boldsymbol{f}(\boldsymbol{x}_i; \mathcal{D}_{(-i)})$ and $C(\boldsymbol{f}_i) \equiv C(\boldsymbol{x}_i)$.

According to a conventional pattern classifier's design, W is found as a

minimizer of the following expected loss:

$$L(W) = \sum_{k=1}^{K} \int P_k \, l_k(\boldsymbol{f}; W) \, p(\boldsymbol{f} \mid C(\boldsymbol{f}) = k) \, d\boldsymbol{f}, \tag{10}$$

where P_k is the prior probability of class k and $l_k(\boldsymbol{f}; W)$ denotes the *loss* caused by misclassification of observation $\boldsymbol{f}$ when $C(\boldsymbol{f}) = k$. Note that loss l_k depends on $\boldsymbol{\alpha}^{(1)}, \ldots, \boldsymbol{\alpha}^{(K)}$ because a classification with a combined classifier is done by a decision rule as: $C(\boldsymbol{x}) = k$ if $f_{com}^{(k)}(\boldsymbol{x}) = \max_j f_{com}^{(j)}(\boldsymbol{x})$.

Since we do not have any knowledge about the distribution of $\boldsymbol{f}$, we cannot directly minimize $L(W)$. In practice, we minimize the following empirical average loss using $\mathcal{D}'$:

$$L'(W) = \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} l_k(\boldsymbol{f}_i; W) \, 1(C(\boldsymbol{f}_i) = k), \tag{11}$$

where $1(\mathcal{U})$ is 1 if $\mathcal{U}$ is true; 0 otherwise.

Minimum Classification Error Discriminant

The most popular loss function is the *zero-one* loss having 1 for misclassification and 0 for correct classification. This loss function is discontinuous at the decision boundary. To derive a gradient algorithm, a smoothed loss function [10] in the form

$$\begin{aligned} l_k(\boldsymbol{f}; W) &= l_k(d_k(\boldsymbol{f}; W)) \\ &= 1/\left(1 + \exp^{-\xi(d_k(\boldsymbol{f}; W) + \tau)}\right) \end{aligned} \tag{12}$$

is suitable. Here, $d_k(\boldsymbol{f}; W)$ is the *misclassification measure* which enumerates how likely $\boldsymbol{f}$ is misclassified as another class. The introduction of a misclassification measure to the loss function originated with Amari [9] and Juang & Katagiri [10]. We employ Juang & Katagiri's definition:

$$d_k(\boldsymbol{f}; \mathbf{W}) = -\boldsymbol{\alpha}^{(k)^T} \boldsymbol{f} + \left(\frac{1}{K-1} \sum_{j, j \neq k} (\boldsymbol{\alpha}^{(j)^T} \boldsymbol{f})^{\eta} \right)^{1/\eta}. \tag{13}$$

Here η is a positive constant. Note that when $\eta \to \infty$, (13) becomes

$$d_k(\boldsymbol{f}; \mathbf{W}) = -\boldsymbol{\alpha}^{(k)^T} \boldsymbol{f} + \max_{j \neq k} \boldsymbol{\alpha}^{(j)^T} \boldsymbol{f}. \tag{14}$$

Clearly, $d_k \leq 0$ means correct classification and $d_k > 0$ indicates misclassification. The above $l_k(d)$ is a smoothed version of the conventional *zero-one* loss, indeed $l_k(d)$ monotonically approaches 0 (1) as d decreases (increases).

With these definitions, $L'(W)$ is now well-defined. Consequently, for the initial estimate $W(0)$, the weight matrix W can be iteratively estimated by using the following *probabilistic descent algorithm.*

$$W(t+1) = W(t) - \epsilon(t) U \sum_{k=1}^{K} 1(C(\boldsymbol{f}) = k) \left. \frac{\partial l_k(\boldsymbol{f}; W)}{\partial W} \right|_{W=W(t)}. \quad (15)$$

Here,

$$\frac{\partial l_k}{\partial \mathbf{W}} \equiv \left(\frac{\partial l_k}{\partial \boldsymbol{\alpha}^{(1)}}, \ldots, \frac{\partial l_k}{\partial \boldsymbol{\alpha}^{(K)}} \right) \in \mathcal{R}^{KM \times K}. \quad (16)$$

U is a small positive definite matrix, $W(t)$ is the estimate at the tth step. From the theory of stochastic approximation [12], it is guaranteed that $W(t)$ will converge to at least a local optimum solution if $\epsilon(t)$ satisfies the following conditions:

$$\lim_{t \to \infty} \epsilon(t) = 0, \quad \sum_{t=1}^{\infty} \epsilon(t) = \infty, \quad \text{and} \quad \sum_{t=1}^{\infty} \epsilon(t)^2 < \infty.$$

One can see that the decision rule is incorporated into the overall cost function $L'(W)$. Therefore, we can obtain linear weights with which a better combined classifier can be constructed for achieving the minimum misclassification probability.

EXPERIMENTS

Two Class Problem

We empirically compared our approach with the stacked regression approach using two class problems for simplicity. The training data set $\mathcal{D} = \{\boldsymbol{x}_i, C(\boldsymbol{x}_i),\ i = 1, \ldots, 100\}$ was artificially generated from the following two-dimensional normal distribution:

Class 1: $\mathcal{N}\left((-0.5\ 0.5)^T, \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix}\right) + \mathcal{N}\left((2.0\ -2.0)^T, \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix}\right)$,

Class 2: $\mathcal{N}\left((0.7\ -0.7)^T, \begin{pmatrix} 1 & 0.6 \\ 0.6 & 1 \end{pmatrix}\right)$.

In the two class problems, one discriminant function for each class is sufficient, because by assuming $0 \leq f_{com}(\boldsymbol{x}) \leq 1$, the decision rule can be written as $C(\boldsymbol{x}) = 1$ if $f_{com}(\boldsymbol{x}) > 0.5$; $C(\boldsymbol{x}) = 2$ otherwise. Thus, (2) is reduced to $f_{com}(\boldsymbol{x}) = \boldsymbol{\alpha}^T \boldsymbol{f}(\boldsymbol{x}; \mathcal{D})$. Therefore, the simple averaging method ($\boldsymbol{\alpha} = (1/3.1/3, 1/3)^T$) was also compared. In this experiment, as individual discriminant functions, we employed feedforward neural network classifiers consisting of two input units, H hidden units, and one output unit. We set $H = 3, 9, 15$ and therefore

$$W = \boldsymbol{\alpha} = (\alpha_1, \alpha_2, \alpha_3)^T \quad \text{and} \quad \boldsymbol{f}(\boldsymbol{x}; \mathcal{D}) = (f_1(\boldsymbol{x}; \mathcal{D}),\ f_2(\boldsymbol{x}; \mathcal{D}),\ f_3(\boldsymbol{x}; \mathcal{D}))^T.$$

Moreover, in this case, (13) was modified as

$$d_1(\boldsymbol{f}; W) = 0.5 - \boldsymbol{\alpha}^T \boldsymbol{f} \quad \text{and} \quad d_2(\boldsymbol{f}; W) = \boldsymbol{\alpha}^T \boldsymbol{f} - 0.5.$$

We set $\xi = 10$ and $\tau = 0$.

Figure 1 shows the classification results obtained by three individual neural network classifiers. As one can easily guess, the simplest and the most complex class boundaries were obtained when $H = 3$ and $H = 15$, respectively. The training and test errors obtained by each classifier are shown in Table 1. A test data set consisting of 3000 points per class was also artificially generated independently of $\mathcal{D}$ from the above distributions.

Figure 2 shows classification results obtained by combined classifiers with different weighting methods, *i.e.*, simple averaging (Method 1), stacked regression (Method 2), and the proposed method (Method 3). Table 2 compares the classification performance of the three methods. The estimated weights by Method 2 were $\boldsymbol{\alpha}_{Method2} = (0.18\ \ 0.49\ \ 0.33)^T$, while $\boldsymbol{\alpha}_{Method3} = (0.55\ \ 0.16\ \ 0.29)^T$ by Method 3. For Method 1, $\boldsymbol{\alpha}_{Method1} = (0.33\ \ 0.33\ \ 0.33)^T$. Since a classifier with $H = 3$ is thought to produce the best classification performace, intuitively $\boldsymbol{\alpha}_{Method3}$ (the first component value is much larger than the others) can be thought of as the best.

Table 2 indicates that the stacked regression approach did not work well (it was worse than the simple averaging method) and that the proposed method produced the smallest classification error among the three methods. Moreover, looking at Tables 1 and 2, one can see that Methods 1 and 2 certainly improved the classification performance, but the test errors obtained from the combined classifiers were larger than that of the best individual classifier (*i.e.*, $H = 3$). On the other hand, in the case of the proposed method, the classification performance of the combined classifier was better improved than those of all individual classifiers, at least in this experiment. This can be seen in Figure 2 where the class boundaries obtained by the proposed method appear to be a smoothed collection of locally desired boundaries obtained by individual classifiers.

Real World Data

We also applied our combination method to a handwritten digit recognition problem (10 class problem) as a real-world case. The training and test data consisted of 200 points/class. Each data point was a 16 dimensional real vector. In this experiment, we used three neural networks with different weight decay parameters ($\lambda = 0.6,\ 0.3,\ 0.15$). The number of hidden units was $H = 20$.

Table 3 shows the misclassification error obtained by each network. As well known, when λ was too large (small), the obtained class bounundaries under- (over-) fitted to the trainig data. Performing an exhaustive line search, we found that $\lambda = 0.3$ produced the smallest test error.

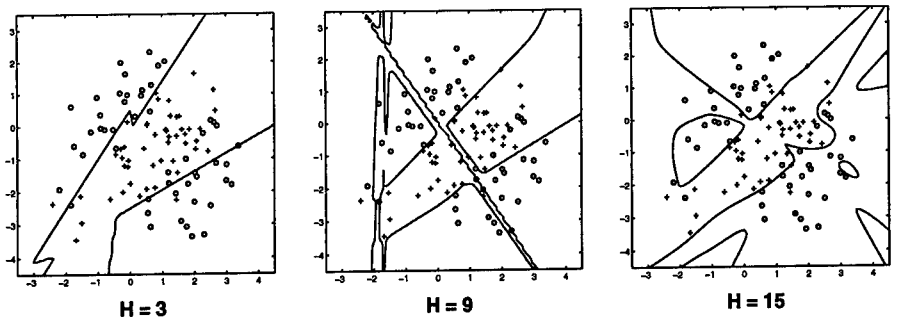


Figure 1: Classification results of neural network classifiers. H is the number of hidden units. $\circ$ and $+$ are class 1 and class 2 sample points, respectively. The solid lines denote class boundaries.

Table 1: Classification performance of individual neural network classifiers. H is the number of hidden units.

	$H = 3$	$H = 9$	$H = 15$
Training error (%)	13.0	11.0	13.0
Test error (%)	17.0	23.0	22.9

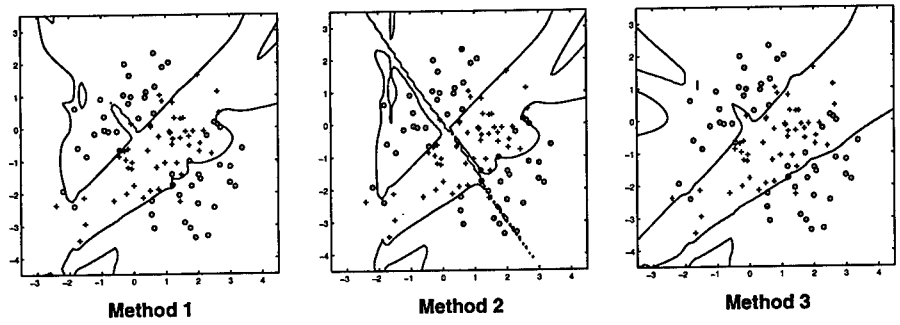


Figure 2: Results of combined classifiers by simple averaging (Method 1), the MSE criterion (Method 2), and the MCE criterion (Method 3).

Table 2: Classification performance of three combining methods: simple averaging (Method 1), stacked regression (Method 2), and the proposed method (Method 3).

	Method 1	Method 2	Method 3
Training error (%)	12.0	12.0	12.0
Test error (%)	18.3	20.6	16.0

Table 3: Classification performance of individual neural network classifiers. λ is the weight decay parameter.

	$\lambda = 0.6$	$\lambda = 0.3$	$\lambda = 0.15$
Training error (%)	5.5	3.9	3.0
Test error (%)	8.8	8.1	8.6

Table 4: Classification performance of combined classifiers.

	MSE	MCE
Training error (%)	2.7	3.4
Test error (%)	8.9	7.6

The results for combined classifiers are compared in Table 4. One can see that the test error obtained by the MCE criterion was less than that by the MSE criterion. Moreover, the combined classifier using the MCE criterion outperformed the best single classifier ($\lambda = 0.3$).

CONCLUSION

We have presented a new way for linearly combining discriminant-based classifiers using the minimum classification error discriminant. One important point is to reformulate the weight estimation problem in combination as the design problem of a linear classifier in an $\boldsymbol{f}$-space. This also made us understand why the stacked regression (MSE) approach is not suitable for the minimum classification error objective, and motivated us to invent our new approach suitable for the objective. In our approach, like in the stacked regression approach, cross-validated stacked data is used to estimate the linear weights. In this sense, we may call our approach *stacked classification.*

In our experiments using combined neural network classifiers, we confirmed the potential advantage of our method. Some of the future research issues involve the classifier selection. In neural network classifiers, it would be better to combine networks with different weight decay parameters than those with different number of hidden units. However, how to select the weight decay parameters in combination is unsettled in this paper. That is, we selected weight decay parameters in some ad hoc manner. This practically important problem remains as a future task.

References

[1] Wolpert D. H., "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, 1992.

[2] Perrone M. P., "Improving regression estimates: Averaging methods for

variance reduction with extensions to general convex measure optimization," *PhD Thesis*, Brown University. 1993.

[3] Breiman L., "Bagging predictors," *Univ. of California, Dept. of Stat.*, TR 421, 1994.

[4] Tresp V. and Taniguchi M., "Combining estimators using non-constant weighting functions," Tesauro G., *et al.* eds., *Advances in Neural Information Processing Systems 7*, MIT Press, Cambridge MA, pp. 419–426, 1995.

[5] Krogh A. and Vedelsby J., "Neural networks ensembles, cross validation, and active learning," Tesauro G., *et al.* eds., *Advances in Neural Information Processing Systems 7*, MIT Press, Cambridge MA, pp. 231–238, 1995.

[6] Tumer K. and Ghosh J., "Analysis of decision boundaries in linearly combined neural classifiers," *Pattern Recognition*, vol. 29, no. 2, pp. 341–348, 1996.

[7] Ueda N. and Nakano R., "Generalization error of ensemble estimators," *Int. Conf. on Neural Networks*, pp. 90–95., 1996.

[8] Breiman L., "Stacked regressions," *Machine Learning*, vol. 24, pp. 49–64, 1996.

[9] Amari S., "A theory of adaptive pattern classifiers," *IEEE Trans. Elec. Comput.*, vol. 16, pp. 299–307, 1967.

[10] Juang B. H. and Katagiri S., "Discriminant learning for minimum error classification," *IEEE Trans. Signal Proc.*, vol. 40, no. 12, 1992.

[11] Duda R. O. and Hart P. E., "Pattern classification and scene analysis," chap. 5., John Wiley & Sons, 1973.

[12] Blum J. R., "Approximation methods which converge with probability one," *Annals of Mathematics and Statistics*, vol. 25, pp. 382–386, 1957.

CROSS-ENTROPY BASED PRUNING OF THE HIERARCHICAL MIXTURES OF EXPERTS

C.C.Whitworth and V.Kadirkamanathan
Department of Automatic Control and Systems Engineering,
The University of Sheffield,
Mappin Street,
Sheffield,
S1 3JD, UK

The paper presents a pruning scheme for the hierarchical mixtures of experts (HME), which is a hierarchical and tree-like modular neural network trained using the EM-algorithm. The pruning scheme is in the style of CART's pruning scheme, and consists of using cross-entropy to select and cut out sub-trees of the HME to create a series of nested HMEs. The right sized HME can then be selected by using cross-validation. Experiments are carried out to demonstrate the successful operation of the scheme.

1. INTRODUCTION

The class of modular networks called the hierarchical mixtures of experts (HME) was first introduced by [1]. The HME is divide-and-conquer method, that does not suffer from the slow learning problems of single neural networks, but neither does it suffer from the large bias of fast-learning tree methods like the Classification and Regression Tree (CART) [2]. This paper is concerned with determining the correct size of the HME, which is crucial as it is concerned with obtaining the optimal bias-variance dilemma [3]. Breiman et al, [2] uses a pruning scheme on the CART after it has grown to an excessively large size, and determines the structure of the tree. In this paper the same idea is applied to the HME to determine its structure, rather than using regularisation [4] or stopping training early to control the complexity [1]. Although [4] developed a pruning scheme for the HME, it uses a threshold to select the sub-tree, and data can be accumulated down the sub-tree at a later time step. The method presented here does not rely on a threshold and permanently prunes the sub-trees of the HME.

The paper begins by introducing the HME and its training in section 2. In section 3 the pruning and merging scheme is explained, and the results obtained presented in section 4. Finally conclusions and discussions are given in section 5.

0-7803-4256-9/97/$10.00

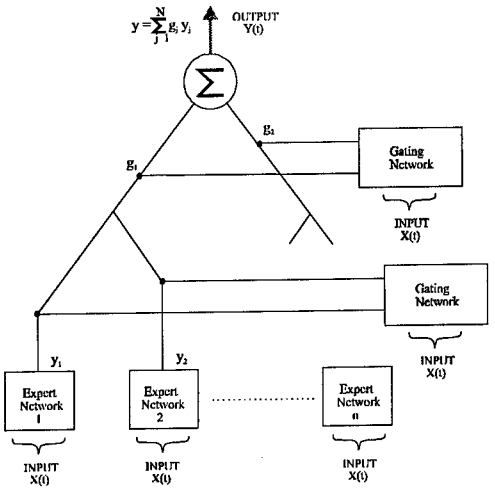


Figure 1 The Hierarchical Mixtures of Experts

2. HIERARCHICAL MIXTURES OF EXPERTS (HME)

The HME is a modular neural network, which consists of several expert networks assigned to different parts of a task by a hierarchy of gating networks (Figure 1). The gating networks weight the outputs of the expert networks, $y_i^{(t)}$, to produce a prediction of the output given by:

$$\hat{y}^{(t)} = \sum_k g_k^{(t)} \left[\sum_i g_{i/k}^{(t)} y_i^{(t)} \right] \qquad (1)$$

$$= \sum_k P\left[m_k / x^{(t)}, \phi_k\right] \left\{ \sum_j P\left[m_j / x^{(t)}, \eta_j\right] E\left[y^{(t)} / x^{(t)}, m_k, m_j, \theta_j\right] \right\}$$

where the sum is over each non-terminal node m_i, and m_k, and θ_i, η_i, ϕ_k are the parameters of the expert and gating networks. $y_i^{(t)}$ and $g_i^{(t)}$ are the notational shorthands for $y_i[x^{(t)}]$ and $g_i[x^{(t)}]$ respectively, and are the individual samples of y_i and g_i. $g_i^{(t)}$ is the gating factor or output of the gating network after it has been passed through the binomial logistic function:

$$g_i^{(t)} = \frac{exp\left[-s_i^{(t)}\right]}{\sum_{n=1}^{2} exp\left[-s_n^{(t)}\right]} \qquad (2)$$

where $s_i^{(t)}$ is the *ith* output of the gating network.

To train the HME, the EM-algorithm is used [5]; it is an iterative optimisation method that ideally suits the modular structure of the HME. The EM-algorithm for the HME works by iterating between estimating the posterior probabilities,

$h_i(t)$ of the gating factors, and training the expert and gating networks. The posterior probabilities for the lowest level gating networks just above the expert networks in a two-level HME are:

$$h_{k/i}^{(t)} = \frac{g_{k/i}^{(t)} P_{ki}^{(t)}}{\sum_k g_{k/i}^{(t)} P_{ki}^{(t)}} \tag{3}$$

and the posterior probabilities in the top-level gating network:

$$h_i^{(t)} = \frac{g_i^{(t)}\left[\sum_k g_{k/i}^{(t)} P_{ki}^{(t)}\right]}{\sum_i g_i^{(t)}\left[\sum_k g_{k/i}^{(t)} P_{ki}^{(t)}\right]} \tag{4}$$

P_k is the conditional likelihood for each expert network:

$$P_k(y/x) = \frac{1}{(2\pi)^{d/2}|R_k|^{1/2}} \exp\left\{-\frac{1}{2}\left[y - \hat{y}_k^{(t)}\right]^T R_k^{-1}\left[y - \hat{y}_k^{(t)}\right]\right\} \tag{5}$$

where y is the target output, R_k the covariance matrix for the *kth* expert, d its dimension, and $\hat{y}_k$ is the estimation of the output by the *kth* expert.

For training the linear expert networks within the iteration of the EM-algorithm, weighted-least-squares is used:

$$w_k^T = \left(XHX^T\right)^{-1} XHY^T \tag{6}$$

where X is the matrix of explanatory variables:

$$X = \begin{bmatrix} x(t-1) & x(t-2) & \cdots & x(t-1-T) \\ x(t-2) & x(t-3) & & x(t-T-2) \\ \vdots & \vdots & & \\ x(t-n) & x(t-n-1) & \cdots & x(t-n-T) \end{bmatrix} \tag{7}$$

and Y is the matrix of regressands:

$$Y = \begin{bmatrix} y(t) & y(t-1) & \cdots & y(t-T) \end{bmatrix} \tag{8}$$

and H is the diagonal matrix of posterior probabilities across time, divided by the standard deviation:

$$H = diag\left[\frac{h_{k,1}^{(t)}}{s_{k/1}} \; \frac{h_{k,1}^{(t)}}{s_{k/1}} \; \ldots \frac{h_{k,1}^{(t)}}{s_{k/1}}\right] \tag{9}$$

However, a non-linear optimisation method like Iterative Weighted-Least-Squares (IRLS) [1] is used for training the gating networks, even when the gating network itself is linear, because of the binomial logistic function.

3. PRUNING SCHEME

The pruning scheme proposed consists of training a large HME until convergence, pruning out the worst sub-tree in the HME, and then reiterating the training-pruning cycle until there is only one expert left on the HME. The HME with the smallest validation error is then selected out of the series of HMEs created by the pruning.

The criterion for selecting the worst sub-tree, is that the root gating network of the sub-tree has the largest cross-entropy:

$$-\sum_{j=1}^{I} h_i^{(t)} log\left[g_i^{(t)}\right] + \left[1 - h_i^{(t)}\right] log\left[1 - g_i^{(t)}\right] \tag{10}$$

The reason for using cross-entropy is because it is a combination of entropy on the posterior, $h_i^{(t)}$ plus the Kullback-Leibler number [6]:

$$-\sum_{j=1}^{I} h_i^{(t)} log\left[h_i^{(t)}\right] + \left[1 - h_i^{(t)}\right] log\left[1 - h_i^{(t)}\right] - \sum_{j=1}^{I} h_i^{(t)} log\left[\frac{g_i^{(t)}}{h_i^{(t)}}\right] + \left[1 - h_i^{(t)}\right] log\left[1 - \frac{g_i^{(t)}}{h_i^{(t)}}\right] \tag{11}$$

The Kullback-Leibler number is a measure of how close $g_i^{(t)}$ is to $h_i^{(t)}$ over all the samples, and is zero when they are identical. An alternative view, is that it represents the amount of information loss that has occurred by the gating network's attempt to learn the posterior. The entropy measures how sharply the posterior, $h_i^{(t)}$ of the gating divides the input space. When the gating network is selected for pruning, it is chosen because it splits up the input space the smoothest, but has also captured the posterior information.

Once the worst sub-tree has been selected, it can then be pruned out, and a new expert network added in its place. The simplest way to initialise the new expert is by randomly generating the weights, but it can cause instability in the training after the pruning. A better method is merging or the weighted average of the expert's weights in a sub-tree, found by feeding the weights of the experts into the sub-tree instead of inputs, and averaging over all samples:

$$w_{new} = \sum_{t=1}^{T}\left(\sum_{i=1}^{I} g_i^{(t)}\left\{\sum_{l=1}^{L} g_{l|i}^{(t)}\left[\sum_{k=1}^{K} g_{k|l,i}^{(t)} w_{k|l,i}\right]\right\}\right) \tag{12}$$

Initialisation of variance is also important in the merging scheme, and all experts' standard deviations should be reinitialised to a large value so that they can compete fairly after merging.

The problem with the merging scheme is that it can not be performed if the experts are non-linear. It also does not produce an expert that is a mean-squares fit to the data in its local region, even when they are linear, and so further training of the whole HME will still have to be performed after the merging.

The freezing method overcomes the problems of merging, by freezing all the weights and standard deviations in the HME, except for the new expert's weights and standard-deviation. The new expert's weights and standard deviation are then trained iteratively, between the expert's weights and its standard deviation as in weighted regression. If the experts are linear the weighted regression becomes an iterated weighted-least-squares, and the weights are obtained by:

$$w_k(n)^T = \left(XGX^T\right)^{-1} XGY^T \tag{13}$$

where X and Y are defined as in (7) and (8) and G is the weighting matrix:

$$G = diag\left[\frac{g_{k,i}^{(t)}}{s_{k|i}} \frac{g_{k,i}^{(t-1)}}{s_{k|i}} \cdots \frac{g_{k,i}^{(t-T)}}{s_{k|i}}\right] \tag{14}$$

and $g_{k,i}$ is the multiplication of all the conditional gating factors, $g_{k|i}$'s down the path to the expert network:

$$g_{k\ldots l,i}^{(t)} = g_i^{(t)} g_{l|i}^{(t)} \cdots g_{k|\ldots l,l}^{(t)} \tag{15}$$

The standard deviation, σ_k is:

$$\sigma_k = \sqrt{\frac{\sum_{t=1}^{T} g_{k|l,i}^{(t)}\left[y^{(t)} - f_k^{(t)}\right]^2}{\sum_{t=1}^{T} g_{k|l,i}^{(t)}}} \tag{16}$$

The gating factor, g_i is used to weight the least-squares to make sure that the new expert is only fitted in the local region covered by the previous two experts that have been replaced. The standard deviation has to be estimated iteratively in the

weighted regression because it is unknown a priori, and is not necessarily the same as the previous two experts.

After pruning, the optimal size of tree is then selected by validation. This validation can be on a single validation set using one series of pruned HMEs. Alternatively, it can be done by selecting the majority-optimal sized HME from n separate series of pruned HMEs trained on n separate cross-validation sets (n-fold cross-validation). The majority-optimal-sized HME is the HME that minimises its cross-validation error the most times.

4. EXPERIMENTAL RESULTS

The experiments use the Sunspot time-series (a benchmark time-series), which is the yearly activity of the sun from 1700 to 1979, and tackled by [7], using neural networks. The actual result of 10-fold cross-validation for one trial is shown in each column of (Table 1) as one trial. Each trial shows that the two-expert HME predominates. Trials two and three use the same initial weights for the HMEs, so that the freezing and merging methods can be compared; the freezing method is more consistent. 10-fold cross-validation was used rather than 5-fold cross-validation which was found to be inconsistent. The gating and expert networks are all linear, with lag vectors of 12 steps as the input. A lag vector of 12 steps is suitable because the periodicity of the time-series is approximately 12 years, and for the comparison with other benchmark results. The HME was used to generate one-step ahead prediction and the normalised mean-squared error (NMSE). NMSE as defined by [7], is the mean-square error normalised by the variance from the whole of the sunspot data (1700-1979).

TABLE 1

	1st trial merging	2nd trial merging	2nd trial freezing
2 Expert Networks	4	3	4
3 Expert Networks	3	3	1

The number of times different sized networks are selected by the cross-validation and pruning in three separate cross-validation trials. Each trial represents 10 pruning-training cycles trained by 10-fold cross-validation. The figures quoted are the number of times that a particular size of HME minimises its cross-validation error in a trial. Results for any HME larger than 3 experts in size are not illustrated in the table, because the pruning and cross-validation never selects these sizes more than once in a given trial.

To compare the performance of the different HME structures, the HMEs produced by the pruning-scheme were then trained on the whole of the training data, and their error-rates compared on the training and two validation sets. In (Figure 2) both the average, minimum and maximum NMSE is worst for the 2-gate HME. However, the 2-gate NMSE performs the best on the two test sets (Figure 3), demonstrating that the pruning-scheme has selected the best size of HME.

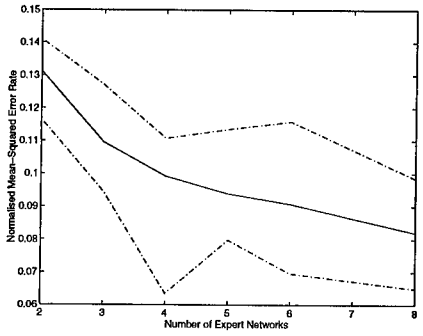


Figure 2 The NMSE of the different sized HMEs on the training set (sunspot 1700-1920). '-' is the average NMSE, and '-.' is the maximum and minimum NMSE.

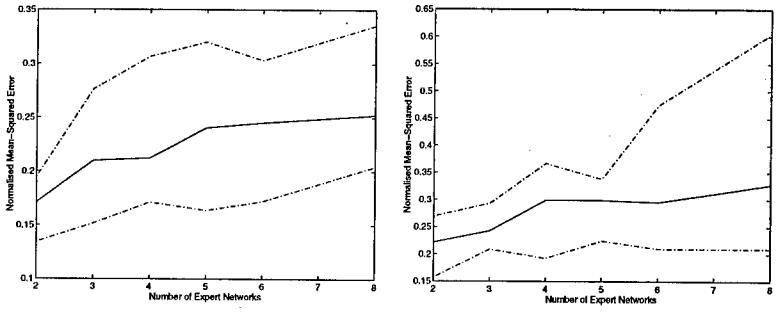


Figure 3 The NMSE of different sized HMEs on the test sets **(a)** 1921-1955 **(b)** 1956-79. '-' is the average NMSE, and '-.' the minimum or maximum NMSE.

The NMSE for our pruning scheme is lower on the (1956-79) test set than the other methods (MLP, TAR etc), and nearly equal with its own result on the (1921-1955) test set (Table 2). This is despite the NMSE being nearly twice as large as the NMSE on the training set and the first test set (1921-55) when compared with the other methods. The pruned HME developed here generalises better across both test sets.

TABLE 2

Method of Training	Training Set (1700-1920) NMSE	Test Set 1 (1921-1955) NMSE	Test Set 2 (1956-1979) NMSE
MLP	0.0862	0.086	0.35
TAR	0.097	0.097	0.28
HME	0.061	0.089	0.27
Linear	0.0984	0.184	0.366
HME pruning	0.117	0.190	0.208

The results of the HME pruning, when compared with other results trained and tested on the same data: the MLP, TAR and Waterhouse's regularised HME (with error-bars). The results are given in the mean-square error normalised by the variance from the whole data (NMSE)

5. DISCUSSION AND CONCLUSION

The pruning scheme results in providing the best generalising HME, with regard to NMSE on the whole Sunspot data. It saves on some of the computational burden that would be required if each size and structure of HME was selected and trained separately. The saving is despite the pruning-scheme using the cross-validation itself with its high computational burden, because the pruning process eliminates many of the possible HME structures that straight cross-validation without pruning uses. Of the two methods of initialising expert networks, the freezing method is superior to the merging method, as it provides consistent results.

Pruning is also a compliment to the growing method. In the CART, the algorithm grows the tree, then prunes back, so that the input space is divided as far as possible, and then the weakest branches can be pruned off. In this paper the growing idea has been approximated by having a larger-than-necessary HME, but currently a growing method to compliment the pruning method is being investigated.

REFERENCES

[1] M.I. Jordan and R.A. Jacobs, "Hierarchical Mixtures of Experts and the EM Algorithm", **Neural Computation**, vol. 6, no. 2, pp. 181-214, March 1994.

[2] L. Breiman, J.H. Friedman, R.A. Olshen and C.J. Stone, **Classification and Regression Trees**, London: Chapman and Hall, 1993.

[3] S. Geman, E. Bienenstock, and R. Doursat, "Neural Networks and the Bias/Variance Dilema", **Neural Computation**, vol. 4, no. 1, pp. 1-58, January 1992.

[4] S.R. Waterhouse and A.J. Robinson, "Pruning and Growing Hierarchical Mixtures of Experts", presented at the 4th International Conference on Artificial Neural Networks, Cambridge, UK, June 26-28, 1995.

[5] A.P. Dempster, N.M. Laird, and D.B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm", **Journal of the Royal Statistical Society: Series B**, vol. 39, no.1, pp. 1-38, December 1976.

[6] S. Kullback, **Information Theory and Statistics**, New York: Dover, 1969, Wiley, 1959.

[7] A.S. Weigend, B.A. Huberman, and D.E. Rumelhart, "Predicting the Future: a Connectionist Approach", **International Journal of Neural Systems**, vol. 1, no.3, pp. 193-209, 1990.

BLIND SIGNAL SEPARATION

Invited Lecture

Blind separation of noisy mixtures

Jean-Francois Cardoso

CNRS/ENST
France

Blind source separation is the problem of recovering independent signals from their mixtures when only the mixtures are observed. In its simplest form, the data model is $\mathbf{x}=\mathbf{A}\mathbf{s}$ where $\mathbf{s}$ is a vector of independent sources, matrix $\mathbf{A}$ is the unknown `mixing matrix' and $\mathbf{x}$ is the vector of observations: the `mixtures'. It is fair to say that this problem is now well understood. In this presentation we consider the problem of recovering the source signals from (noisy) observations. This is modeled as $\mathbf{x}=\mathbf{A}\mathbf{s}+\mathbf{n}$ where vector $\mathbf{n}$ represents an additive noise, independent from the signals. There are many issues related to the noisy problem, which make it significantly different from the noise-free problem. This talk reviews what is known about the noisy case and presents new results. The following points are addressed:

- What is the optimal filter to recover unknown sources from noisy observations?
- How to use high order information to define contrast functions which remain consistent in the presence of noise.
- What does the likelihood have to say in a noisy context?
- Learning noisy mixtures with the EM (Expectation-Maximization) algorithm and its stochastic variants
- The tricky continuity between noisy and noise-free optimal algorithms.
- Achievable performance in the presence of noise.
- When is a noise-free algorithm appropriate to deal with noisy observations?
- Is it really worth to care about observation noise?

ONE-UNIT CONTRAST FUNCTIONS FOR INDEPENDENT COMPONENT ANALYSIS: A STATISTICAL ANALYSIS

Aapo Hyvärinen

Helsinki University of Technology
Laboratory of Computer and Information Science
P.O. Box 2200, FIN-02015 HUT, Finland
Email: aapo.hyvarinen@hut.fi

ABSTRACT

The author introduced previously a large family of one-unit contrast functions to be used in independent component analysis (ICA). In this paper, the family is analyzed mathematically in the case of a finite sample. Two aspects of the estimators obtained using such contrast functions are considered: asymptotic variance, and robustness against outliers. An expression for the contrast function that minimizes the asymptotic variance is obtained as a function of the probability densities of the independent components. Combined with robustness considerations, these results provide strong arguments in favor of the use of contrast functions based on slowly growing functions, and against the use of kurtosis, which is the classical contrast function.

1. INTRODUCTION

Independent Component Analysis (ICA) [1] is a statistical signal processing technique whose main applications are blind source separation, blind deconvolution, and feature extraction. In the simplest form of ICA [2], one observes m scalar random variables $x_1, x_2, ..., x_m$ which are assumed to be linear combinations of n unknown independent components, or ICs, denoted by $s_1, s_2, ..., s_n$. These ICs s_i are assumed to be mutually *statistically independent*, and zero-mean. Arranging the observed variables x_j into a vector $\mathbf{x} = (x_1, x_2, ..., x_m)^T$ and the IC variables s_i into a vector $\mathbf{s}$, the linear relationship can be expressed as

$$\mathbf{x} = \mathbf{A}\mathbf{s} \tag{1}$$

Here, $\mathbf{A}$ is an unknown $m \times n$ matrix of full column rank, called the mixing matrix. The basic problem of ICA is then to estimate both the mixing matrix $\mathbf{A}$ and the realizations of the ICs s_i *using only observations of the mixtures* x_j.

0-7803-4256-9/97/$10.00

Estimation of ICA requires the use of higher-order information, i.e., other information than that contained in the covariance matrix of $\mathbf{x}$. This higher-order information is usually incorporated in the estimation procedures by means of 'contrast' functions based on higher-order cumulants [2, 3]. However, little justification has been provided in the literature for the choice of using higher-order cumulants for the construction of the contrast functions. The main reason for their popularity seems to be that they are easy to analyze mathematically. No statistical or practical arguments in favor of cumulants have been put forth, except for the fact that they may be more resistant to Gaussian noise, because the higher-order cumulants of Gaussian noise vanish.

In this paper, we analyze mathematically a large family of one-unit contrast functions introduced in [4]. The asymptotic variance of the obtained estimators is evaluated, and it is shown that for super-Gaussian ICs, the asymptotic variance is minimized for contrast functions that grow much slower than the 4-th power inherent in the fourth-order cumulant or kurtosis. Furthermore, robustness against outliers also requires slowly growing contrast functions. As most ICs encountered in practice seem to be super-Gaussian, this means that kurtosis may be a rather inadequate contrast function in most cases. For neural learning rules, the results imply that better estimates are usually obtained using (anti-)Hebbian learning functions that are sigmoidal, or even go to zero at infinity. Simulations back up our theoretical arguments.

2. GENERAL ONE-UNIT CONTRAST FUNCTIONS

Consider a linear combination of the observed mixtures x_j, say $\mathbf{w}^T\mathbf{x}$, where the (weight) vector $\mathbf{w}$ is constrained so that $E\{(\mathbf{w}^T\mathbf{x})^2\} = 1$. Many ICA algorithms are based on finding the extrema of the square of the kurtosis $\mathrm{kurt}^2(\mathbf{w}^T\mathbf{x}) = (E\{(\mathbf{w}^T\mathbf{x})^4\} - 3)^2$ of such a linear combination [2, 3]. This can be motivated by information-theoretic arguments: the square of the kurtosis can be shown to approximate the negentropy of $\mathbf{w}^T\mathbf{x}$ [2]. Moreover, it can be proven that the square of the kurtosis of $\mathbf{w}^T\mathbf{x}$ is maximized exactly in the points where the linear combination equals, up to the sign, one of the ICs, i.e., $\mathbf{w}^T\mathbf{x} = \pm s_i$ for some i [3, 5].

This approach was generalized in [4, 6, 7], where it was shown that instead of kurtosis, practically any non-quadratic, well-behaving even function, say G, can be used to construct a contrast function for ICA. Such a general contrast function can be defined as

$$J_G(\mathbf{w}) = [E_{\mathbf{x}}\{G(\mathbf{w}^T\mathbf{x})\} - E_\nu\{G(\nu)\}]^2 \tag{2}$$

where ν is a standardized Gaussian variable. The second term in brackets is a normalization constant that makes J_G equal to zero if $\mathbf{w}^T\mathbf{x}$ has a Gaussian distribution. Clearly, J_G can be considered a generalization of the square of kurtosis, as for $G(u) = u^4$, J_G becomes simply the square of kurtosis of $\mathbf{w}^T\mathbf{x}$. It was shown in [6], using a generalization of the Gram-Charlier expansion, how J_G approximates the negentropy of $\mathbf{w}^T\mathbf{x}$ in the same way as the square of the kurtosis. Furthermore, it was shown in [7] that under weak assumptions,

J_G is locally maximized when $\mathbf{w}^T\mathbf{x} = \pm s_i$, i.e. when the linear combination equals one of the ICs. Therefore, J_G can be used as a contrast function for ICA in the same way as the square of the kurtosis. Note that for simplicity, we shall also refer to G as a contrast function.

Thus, we estimate one IC by solving the following optimization problem:

$$\widehat{\mathbf{w}} = \arg \max_{E\{(\mathbf{w}^T\mathbf{x})^2\}=1} J_G(\mathbf{w}) \tag{3}$$

where in practice the expectations are replaced by sample averages. Note that this boils down to maximizing or minimizing $E\{G(\mathbf{w}^T\mathbf{x})\}$, where the type of extrema searched for depends on the sign of $E_{\mathbf{x}}\{G(\mathbf{w}^T\mathbf{x})\} - E_{\nu}\{G(\nu)\}$. To estimate all the ICs, one needs only to find all the local solutions of this optimization problem. We shall not consider here in detail how to solve this optimization. Two simple methods are possible. First, one can use a simple gradient descent/ascent with a decreasing learning rate, as is considered in more detail in [7]. In that case it may be useful to first whiten (or sphere) the data, which simplifies the constraint to $||\mathbf{w}|| = 1$. A second possibility is the fixed-point algorithm introduced for kurtosis in [8] and generalized for any G in [4]. However, the statistical properties of the estimator defined in (3) do not depend on the method of optimization.

In the following, we shall analyze two fundamental statistical properties of $\widehat{\mathbf{w}}$, which are asymptotic variance and robustness. Though in principle almost any non-quadratic even function G can be used, in practice the performance of different contrast functions may be very different due to limited sample sizes and deviations from the model (1). Therefore, some analysis is needed to provide guidelines on how to choose the function G to obtain a statistically adequate estimator.

3. ASYMPTOTIC VARIANCE

In practice, one usually has only a finite sample of N observations of the vector $\mathbf{x}$. Therefore, the expectations in the definition of J_G are in fact replaced by sample averages. This results in certain errors in the estimator $\widehat{\mathbf{w}}$, and it is desired to make these errors as small as possible. A classical measure of this error is asymptotic (co)variance, which means the limit of the covariance matrix of $\widehat{\mathbf{w}}\sqrt{N}$ as $N \to \infty$. This gives an approximation of the mean-square error of $\widehat{\mathbf{w}}$. Comparison of, say, the traces of the asymptotic variances of two estimators enables direct comparison of the accuracy of two estimators. One can solve analytically for the asymptotic variance of $\widehat{\mathbf{w}}$, obtaining the following theorem:

Theorem 1 *The trace of the asymptotic variance of $\widehat{\mathbf{w}}$ as defined in (3) for the estimation of the independent component s_i, equals*

$$V_G = C(\mathbf{A})\frac{E\{g^2(s_i)\} - (E\{s_i g(s_i)\})^2}{(E\{s_i g(s_i) - g'(s_i)\})^2}, \tag{4}$$

where g is the derivative of G, and $C(\mathbf{A})$ is a constant that depends only on $\mathbf{A}$.

Proof: Making the change of variable $\mathbf{z} = \mathbf{A}^T\mathbf{w}$, the equation defining the optimal solutions $\hat{\mathbf{z}}$ becomes

$$\sum_t \mathbf{s}_t g(\hat{\mathbf{z}}^T\mathbf{s}_t) = \lambda \sum_t \mathbf{s}_t \mathbf{s}_t^T \hat{\mathbf{z}} \tag{5}$$

where $t = 1, .., T$ is the sample index, T is the sample size, and λ is a Lagrangian multiplier.. Without loss of generality, let us assume that $\hat{\mathbf{z}}$ is near the ideal solution $\mathbf{z} = (1, 0, 0, ...)$. Note that due to the constraint $E\{(\mathbf{w}^T\mathbf{x})^2\} = \|\mathbf{z}\|^2 = 1$, the variance of the first component of $\hat{\mathbf{z}}$, denoted by $\hat{z}_1$, is of a smaller order than the variance of the vector of other components, denoted by $\hat{\mathbf{z}}_{-1}$. Excluding the first component in (5), and making the first-order approximation $g(\hat{\mathbf{z}}^T\mathbf{s}) = g(s_1) + g'(s_1)\hat{\mathbf{z}}_{-1}^T\mathbf{s}_{-1}$, where also $\mathbf{s}_{-1}$ denotes $\mathbf{s}$ without its first component, one obtains after some simple manipulations

$$\frac{1}{\sqrt{T}} \sum_t \mathbf{s}_{-1}[g(s_1) - \lambda s_1] = \frac{1}{T} \sum_t \mathbf{s}_{-1}[-\mathbf{s}_{-1}^T g'(s_1) + \lambda \mathbf{s}_{-1}^T]\hat{\mathbf{z}}_{-1}\sqrt{T} \tag{6}$$

where the sample index t has been dropped for simplicity. Making the first-order approximation $\lambda = E\{s_1 g(s_1)\}$, one can write (6) in the form $u = v\hat{\mathbf{z}}_{-1}\sqrt{T}$ where v converges to the identity matrix multiplied by $E\{s_1 g(s_1)\} - E\{g'(s_1)\}$, and u converges to a variable that has a normal distribution of zero mean whose covariance matrix equals the identity matrix multiplied by $E\{g^2(s_1)\} - (E\{s_1 g(s_1)\})^2$. This implies the theorem, since $\hat{\mathbf{z}}_{-1} = \mathbf{B}\hat{\mathbf{w}}$, where $\mathbf{B}$ is the inverse of $\mathbf{A}^T$ without its first row.

Thus the comparison of the asymptotic variances of two estimators of the form in (3), but for two different contrast functions G, boils down to a comparison of the V_G's. In particular, one can use variational calculus to find a G that minimises V_G. Thus one obtains the following theorem:

Theorem 2 *The trace of the asymptotic variance of $\widehat{\mathbf{w}}$ is minimized when G is of the form*

$$G_{opt}(u) = c_1 \log f(u) + c_2 u^2 + c_3 \tag{7}$$

where f is the density function of s_i, and c_1, c_2, c_3 are arbitrary constants.

For simplicity, one can choose $G_{opt}(u) = \log f(u)$. Thus one sees that the optimal contrast function is the same as the one obtained for several units by the maximum likelihood approach [9], or the infomax approach [10]. Almost identical results have also been obtained in [11] for another multi-unit algorithm. Our results treat, however, the one-unit case instead of the multi-unit case, and are thus applicable to estimation of a subset of the ICs, and to blind deconvolution [7].

4. ROBUSTNESS

Another very desirable property of an estimator is robustness against outliers [12]. This means that single, highly erroneous observations do not have much influence on the estimator.

In this paper, we shall treat the question: How does the robustness of the estimator $\hat{\mathbf{w}}$ depend on the choice of the function G? Note that the robustness of $\hat{\mathbf{w}}$ depends also on the method of estimation used in constraining the variance of $\mathbf{w}^T\mathbf{x}$ to equal unity in (3). This is a problem independent of the choice of G. In the following, we assume that this constraint is implemented in a robust way. In particular, we assume that the data is sphered (whitened) in a robust manner, in which case the constraint reduces to $\|\mathbf{w}\| = 1$. Several robust estimators of the variance of $\mathbf{w}^T\mathbf{x}$ or of the covariance matrix of $\mathbf{x}$ are presented in the literature; see [12].

The robustness of the estimator $\hat{\mathbf{w}}$ in (3) can be analyzed using the theory of M-estimators [12]. Without going into technical details, the definition of an M-estimator can be formulated as follows: an estimator is called an M-estimator if it is defined as the solution $\hat{\theta}$ for θ of

$$E\{\psi(\mathbf{x}, \theta)\} = 0 \tag{8}$$

where $\mathbf{x}$ is a random vector and ψ is some function defining the estimator. The estimator $\hat{\mathbf{w}}$ in (3) is an M-estimator. To see this, define $\theta = (\mathbf{w}, \lambda)$, where λ is the Lagrangian multiplier associated with the constraint. Using the Kuhn-Tucker conditions, the estimator $\hat{\mathbf{w}}$ can then be formulated as the solution of equation (8) where $\psi = \psi_J$ is defined as follows (for sphered data):

$$\psi_J(\mathbf{x}, \theta) = \begin{pmatrix} \mathbf{x}g(\mathbf{w}^T\mathbf{x}) + c\lambda\mathbf{w} \\ \|\mathbf{w}\|^2 - 1 \end{pmatrix} \tag{9}$$

where $c = (E_{\mathbf{x}}\{G(\hat{\mathbf{w}}^T\mathbf{x})\} - E_{\nu}\{G(\nu)\})^{-1}$ is an irrelevant constant.

The analysis of robustness of an M-estimator is based on the concept of an infuence function, $\mathrm{IF}(\mathbf{x}, \hat{\theta})$. Intuitively speaking, the influence function measures the influence of single observations on the estimator. It would be desirable to have an influence function that is bounded as a function of $\mathbf{x}$, as this implies that even the influence of a far-away outlier is 'bounded', and cannot change the estimate too much. This requirement leads to one definition of robustness, which is called B-robustness. An estimator is called B-robust, if its influence function is bounded as a function of $\mathbf{x}$, i.e., $\sup_{\mathbf{x}} \|\mathrm{IF}(\mathbf{x}, \hat{\theta})\|$ is finite for every $\hat{\theta}$. Even if the influence function is not bounded, it should grow as slowly as possible when $\|\mathbf{x}\|$ grows, to reduce the distorting effect of outliers.

It can be shown [12] that the influence function of an M-estimator equals

$$\mathrm{IF}(\mathbf{x}, \hat{\theta}) = \mathbf{B}\psi(\mathbf{x}, \hat{\theta}) \tag{10}$$

where $\mathbf{B}$ is an irrelevant invertible matrix that does not depend on $\mathbf{x}$. On the other hand, using our definition of ψ_J, and denoting by $\gamma = \mathbf{w}^T\mathbf{x}/\|\mathbf{x}\|$ the cosine of the angle between $\mathbf{x}$ and $\mathbf{w}$, one obtains easily

$$\|\psi(\mathbf{x}, (\mathbf{w}, \lambda))\|^2 = C_1\frac{1}{\gamma^2}h^2(\mathbf{w}^T\mathbf{x}) + C_2h(\mathbf{w}^T\mathbf{x}) + C_3 \tag{11}$$

where C_1, C_2, C_3 are constants that do not depend on $\mathbf{x}$, and $h(u) = ug(u)$. Thus on sees that the robustness of $\hat{\mathbf{w}}$ essentially depends on the behavior of the function $h(u)$. The slower $h(u)$ grows, the more robust the estimator. However, the estimator cannot be really B-robust, because the γ in the denominator prevents the influence function from being bounded for all $\mathbf{x}$. In particular, outliers that are almost orthogonal to $\hat{\mathbf{w}}$, and have large norms, may still have a large influence on the estimator. These results are stated in the following theorem:

Theorem 3 *Assume that the data* $\mathbf{x}$ *is whitened (sphered) in a robust manner. Then the influence function of the estimator* $\hat{\mathbf{w}}$ *is never bounded for all* $\mathbf{x}$. *However, if* $h(u) = ug(u)$ *is bounded, the influence function is bounded in sets of the form* $\{\mathbf{x} \mid \hat{\mathbf{w}}^T\mathbf{x}/\|\mathbf{x}\| > \epsilon\}$ *for every* $\epsilon > 0$, *where* g *is the derivative of* G.

In particular, if one chooses *a contrast function* $G(u)$ *that is bounded,* h is also bounded, and $\hat{\mathbf{w}}$ is quite robust against outliers. If this is not possible, one should at least choose a contrast function $G(u)$ that does not grow very fast when $|u|$ grows. If, in contrast, $G(u)$ grows very fast when $|u|$ grows, the estimates depend mostly on a few observations far from the origin. This leads to highly non-robust estimators, which can be completely ruined by just a couple of bad outliers. This is the case, for example, when kurtosis is used as a contrast function, which is equivalent to using $\hat{\mathbf{w}}$ with $G(u) = u^4$.

5. CHOOSING THE CONTRAST FUNCTION IN PRACTICE

It is useful to analyze the implications of the theoretical results of the preceding sections by considering the following family of density functions:

$$f_\alpha(x) = C_1 \exp(C_2|x|^\alpha) \tag{12}$$

where α is a positive constant, and C_1, C_2 are normalization constants that ensure that f_α is a probability density of unit variance. For different values of alpha, the densities in this family exhibit different shapes. For $.5 < \alpha < 2$, one obtains a sparse, super-Gaussian density (i.e. a density of positive kurtosis). For $\alpha = 2$, one obtains the Gaussian distribution, and for $\alpha > 2$, a sub-Gaussian density (i.e. a density of negative kurtosis). Thus the densities in this family can be used as examples of different non-Gaussian densities.

Using Theorem 2, one sees that in terms of asymptotic variance, an optimal contrast function for estimating an IC whose density function equals f_α, is of the form:

$$G_{opt}(u) = |u|^\alpha \tag{13}$$

where the arbitrary constants have been dropped for simplicity. This implies roughly that for super-Gaussian (resp. sub-Gaussian) densities, the optimal contrast function is a function that grows *slower than quadratically* (resp. *faster than quadratically*). Next, recall from Section 4 that if $G(u)$

grows fast with $|u|$, the estimator becomes highly non-robust against outliers. Taking also into account the fact that most ICs encountered in practice are super-Gaussian, one reaches the conclusion that as a general-purpose contrast function, one should choose a function G that resembles rather

$$G_{opt}(u) = |u|^{\alpha}, \text{where } \alpha < 2. \tag{14}$$

The problem with such contrast functions is, however, that they are not differentiable at 0 for $\alpha \leq 1$. Thus it is better to use approximating differentiable functions that have the same kind of qualitative behavior. Considering $\alpha = 1$, in which case one has a double exponential density, one could use instead the function $G_1(u) = \log\cosh a_1 u$ where $a_1 > 1$ is a moderately large constant. Note that the derivative of G_1 is then the familiar tanh function (for $a_1 = 1$). In the case of $\alpha < 1$, i.e. highly super-Gaussian ICs, one could approximate the behavior of G_{opt} for large u using a Gaussian function (with a minus sign): $G_2(u) = -\exp(-a_2 u^2/2)$ where a_2 is a constant. The derivative of this function is like a sigmoid for small values, but goes to 0 for larger values. Note that this function also fulfills the condition in Theorem 3, thus providing an estimator that is as robust as possible in this framework. We have found $a_1 = 2$ and $a_2 = 1$ to provide 'good' approximations of G_1 and G_2. Note that there is a trade-off between the precision of the approximation and the smoothness of the resulting objective function.

Thus, we reach the following general conclusion:

- a good general-purpose contrast function is $G(u) = \log\cosh a_1 u$, where $a_1 \geq 1$ is a constant.
- when the ICs are highly super-Gaussian, or when robustness is very important, $G(u) = -\exp(-a_2 u^2/2)$ with $a_2 \approx 1$ may be better.
- using kurtosis is justified only if the ICs are sub-Gaussian and there are no outliers.

In this paper, we have used purely statistical criteria for choosing the contrast function. One important criterion that is completely independent of statistical considerations is computational simplicity. For example, the calculation of the tanh function is rather slow in many environments. The convergence may be speeded up if one uses instead piecewise linear approximations of the derivatives of the contrast functions. In the case of $g(u) = \tanh(a_2 u)$, one may define g so that $g(u) = a_3 u$ for $|u| < 1/a_3$ and $g(u) = \text{sign}(u)$ otherwise, where $a_3 \geq 1$ is a constant. This amounts to using the so-called Huber function [12] as G.

6. SIMULATIONS

We performed simulations in which 3 different contrast functions were used to estimate one IC from a mixture of 4 i.i.d. ICs. The contrast functions used were kurtosis, and the two functions proposed in the preceding section:

log cosh (or G_1) and the Gaussian function (or G_2). The constants were set as suggested in the preceding section. We also used three different distributions of the ICs: uniform, double exponential (or Laplace), and the distribution of the third power of a Gaussian variable. The sample size was fixed at 1000 and the fixed-point algorithm in [4] was used to maximize the contrast function. The asymptotic mean absolute deviations (MAD) between the components of the obtained vectors and the correct solutions were estimated and averaged over 1000 runs for each combination of non-linearity and distribution of IC. MAD was used instead af variance because it is a more robust measure of deviation.

The results in the basic, noiseless case are depicted in Fig. 1. As one can see, the estimates using kurtosis were essentially worse for super-Gaussian ICs. Especially the strongly super-Gaussian IC (cube of Gaussian) was estimated considerably worse using kurtosis. Only for the sub-Gaussian IC, kurtosis was better than the other contrast functions. There was no clear difference between the performances of the contrast functions G_1 and G_2.

Next, the experiments were repeated with added Gaussian noise whose energy was 10% of the energy of the ICs. The results are shown in Fig. 2. This time, kurtosis did not perform better even in the case of the sub-Gaussian density. This result goes against the view that kurtosis would tolerate Gaussian noise well. Indeed, the theoretical arguments supporting that view neglect any finite-sample effects, and may thus have rather limited validity.

No outliers were added in these experiments. Experiments confirming the robustness of the non-linearities proposed in section 5 can be found in [4].

7. CONCLUSION

The problem of choosing the contrast function for ICA was treated. The behavior of a large family of contrast functions, which includes kurtosis as a special case, was analyzed. Combining the results on asymptotic variance and robustness against outliers, it was shown that the use of kurtosis is not justified on statistical grounds, except perhaps for sub-Gaussian independent components. Instead, contrast functions that grow slower than quadratically were found to be better approximations of the optimal ones in most cases. In neural learning rules, this leads, e.g., to the use of tanh-like sigmoids, or functions resembling the derivative of a Gaussian function.

8. REFERENCES

[1] C. Jutten and J. Herault, "Blind separation of sources, part I: An adaptive algorithm based on neuromimetic architecture," *Signal Processing*, vol. 24, pp. 1–10, 1991.

[2] P. Comon, "Independent component analysis – a new concept?," *Signal Processing*, vol. 36, pp. 287–314, 1994.

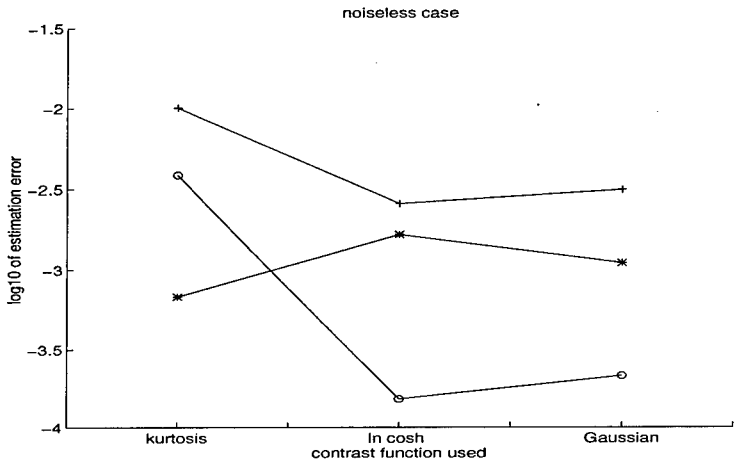


Figure 1: Estimation errors plotted for different contrast functions and distributions of the ICs, in the noiseless case. Asterisk: uniform distribution. Plus sign: Double exponential. Circle: cube of Gaussian.

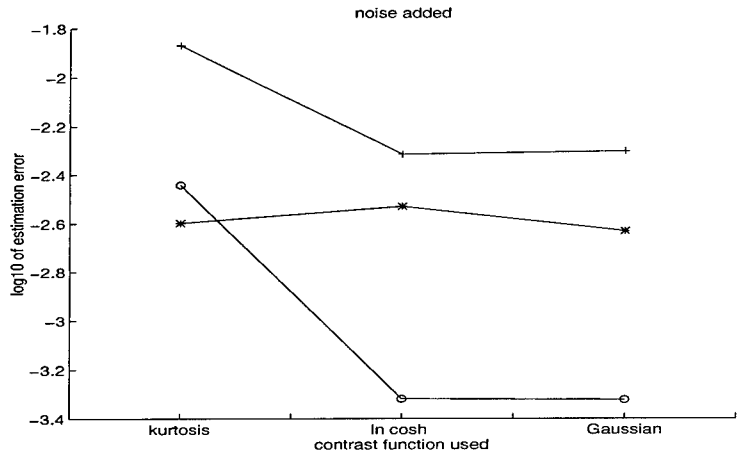


Figure 2: The noisy case. Estimation errors plotted for different contrast functions and distributions of the ICs. Asterisk: uniform distribution. Plus sign: Double exponential. Circle: cube of Gaussian.

[3] N. Delfosse and P. Loubaton, "Adaptive blind separation of independent sources: a deflation approach," *Signal Processing*, vol. 45, pp. 59–83, 1995.

[4] A. Hyvärinen, "A family of fixed-point algorithms for independent component analysis," in *Proc. ICASSP'97*, (Munich, Germany), pp. 3917–3920, 1997.

[5] A. Hyvärinen and E. Oja, "One-unit learning rules for independent component analysis," in *Advances in Neural Information Processing Systems 9 (NIPS*96)*, MIT Press, 1997. To appear.

[6] A. Hyvärinen, "Approximations of differential entropy for independent component analysis and projection pursuit," 1997. Submitted.

[7] A. Hyvärinen and E. Oja, "Independent component analysis by general non-linear hebbian-like learning rules," 1997. Submitted.

[8] A. Hyvärinen and E. Oja, "A fast fixed-point algorithm for independent component analysis," *Neural Computation*, 1997. To appear.

[9] D.-T. Pham, P. Garrat, and C. Jutten, "Separation of a mixture of independent sources through a maximum likelihood approach," in *Proc. EUSIPCO*, pp. 771–774, 1992.

[10] A. Bell and T. Sejnowski, "An information-maximization approach to blind separation and blind deconvolution," *Neural Computation*, vol. 7, pp. 1129–1159, 1995.

[11] J.-F. Cardoso and B. H. Laheld, "Equivariant adaptive source separation," *IEEE Trans. on Signal Processing*, vol. 44, no. 12, pp. 3017–3030, 1996.

[12] F. Hampel, E. Ronchetti, P. Rousseuw, and W. Stahel, *Robust Statistics*. Wiley, 1986.

Feature Extraction Approach to Blind Source Separation

Juan K. Lin
Department of Physics
University of Chicago
Chicago, IL 60637
jk-lin@uchicago.edu

David G. Grier
Department of Physics
University of Chicago
Chicago, IL 60637
d-grier@uchicago.edu

Jack D. Cowan
Department of Math
University of Chicago
Chicago, IL 60637
j-cowan@uchicago.edu

Abstract

Local independent component analysis is formulated as a task involving the extraction of local geometric structure in the joint distribution. Because the geometrical structure of statistical independence is not well captured by statistical descriptions such as moments and cumulants, we use feature detection tools from image analysis to locate the local independent component coordinate system. The resulting approach to source separation can be implemented in real time using conventional image analysis hardware. The generality of this approach is demonstrated by blind source separation of multi–modal sources, and the pseudo–separation of three sources from two mixtures.

1 INTRODUCTION

The blind source separation or independent component analysis (ICA) algorithms of Bell and Sejnowski [2], Pearlmutter and Parra [8], Amari, Cichocki, Yang [1] and Cardoso and Laheld [5] all attempt to find a global coordinate system where the joint distribution takes on a product form. These linear ICA algorithms are all non–local and linear in the sense that non–local information is used and hence only linear mixtures can be separated. They involve stochastic gradient descent of the density estimation parameters on a cost function such as the Kullback-Leibler divergence, and only work when given adequate priors on the joint distribution's parametric form. Recently, the authors directed attention to the intrinsic local structure of source mixtures, and introduced a local aligned equipartition approach [6, 7]. The resulting algorithm performs non–parametric density estimation and blind source separation of non–linear mixtures. In this contribution, we further pursue the

0-7803-4256-9/97/$10.00

local geometric feature approach, but instead use well studied tools from the field of image analysis for local feature extraction. In contrast to density estimation source separation algorithms, we use "edge features" in the density distribution to locate the independent component coordinate system. We demonstrate two dimensional source separation for visual and computation reasons, though the approach is clearly not limited in dimensionality.

2 LOCAL GEOMETRICAL STRUCTURE

In the blind separation problem, we are given input consisting of a non–singular mixture of statistically independent sources, $\{\vec{x}\} = \{A\vec{s}\}$ where the joint distribution in the source frame factorizes: $P_s(\vec{s}) = \prod p_k(s_k)$. First we rewrite the mixing equation in a more suggestive form with the j_{th} column of A denoted by the vector $|a_j\rangle$

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \end{pmatrix} = \begin{pmatrix} |a_1\rangle & |a_2\rangle & \dots & |a_N\rangle \end{pmatrix} \begin{pmatrix} s_1 \\ s_2 \\ \vdots \end{pmatrix}. \tag{1}$$

The column vectors of A define an independent component coordinate system. The independent component basis vectors are in general not normalized and orthogonal. From the mixing equation, it can be seen that the components of the input in the independent component coordinate system are the independent source amplitudes. As an example, for sharply peaked sources, when only one source s_i deviates significantly from its mean, the mixed signal will fall predominantly along a line parallel to $\vec{a_i}$. More generally, high density regions or directions are mapped to other high density regions or directions. In the source frame, the high/low density regions are determined by the location of the extrema of the individual source distributions: $p'_k(s_k) = 0$. Consequently the extremal density directions are parallel to the source directions. Locating the extremal density directions in the mixture frame thus allows for source extraction from the mixture.

3 SOURCE SEPARATION ALGORITHM

The local feature detection source separation algorithm operates in batch mode, and consists of the following steps:

a. Obtain histogram of the joint distribution
The input vector space is partitioned into bins. A straightforward counting of the input data points in each bin gives us the histogram of the distribution.

b. Determine the gradient of the distribution
We convolve the binned data set with a discrete derivative operator of support L to approximate the local gradients of the joint probability distribution.

In practice, we have found that $L = 7$ for our numerics is a good compromise between accuracy of the gradient estimate and appropriate sensitivity to curvature in the data set.

c. Threshold the gradient
We first normalize to the maximum amplitude of the gradient. Then we consider only the regions where the magnitude of the normalized gradient is larger than a certain threshold. These regions are what we term "edges" of the distribution. An extrema density region in the joint distribution will be surrounded by two regions of high gradient with roughly opposite orientations. Thus there are two parallel lines surrounding each extremal direction. Numerically, we found that the directions of these lines could be determined more accurately by considering them separately. This was accomplished by partitioning the large gradient regions according to whether the gradient orientation points in the upper or lower half plane.

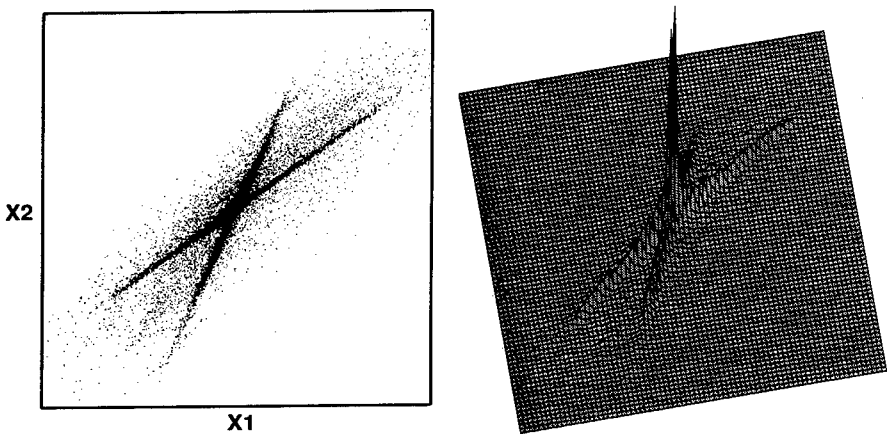


Figure 1: Left: input distribution in the mixture coordinate frame. Right: density histogram mesh plot.

d. Hough transform the edge image
The resulting binary images of the edges of the distribution are Hough transformed for the detection of high gradient lines (see e.g. Jahne [4]). The peaks in the Hough transform, which correspond to the "most popular lines", are located. By inserting the two most popular line orientations into an unmixing matrix, an unmixing transformation which separates the mixture can be obtained.

4 SIMULATION RESULTS

4.1 Separation of unimodal sources

Two audio files consisting of spoken Japanese phrases were linearly mixed by a random mixing matrix. The input distribution consisting of 34934 points was histogrammed into 80×80 bins, as shown in Figure 1. The plot clearly shows the geometrical structure discussed in Section 1. The normalized gradient of the histogram is computed and thresholded at five percent of the maximum value, giving us the "edges" of the joint distribution. Figure 2 shows the "edge" regions corresponding to large gradient regions pointing in the lower half plane. The Hough transforms of the binary images are shown alongside. From the two most popular line orientations, An unmixing matrix was constructed from the two most popular line orientations. Left multiplying the mixing matrix with the unmixing matrix, we find the resulting mixture to signal ratio of the two outputs to be 0.95 and 3.8 percent respectively.

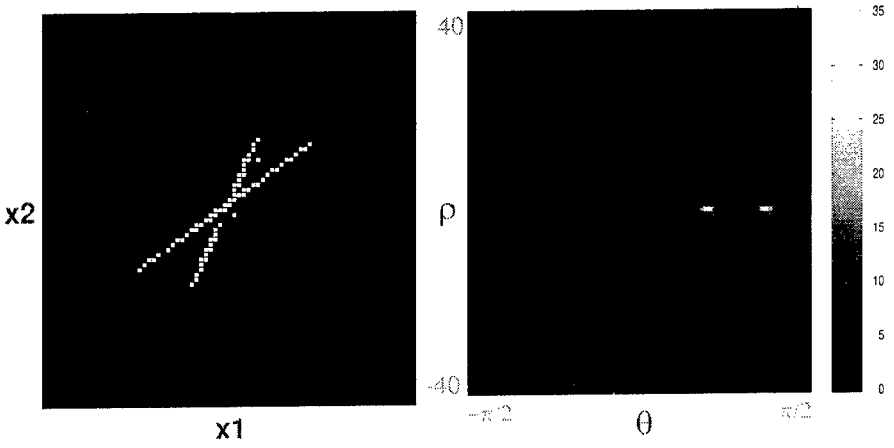


Figure 2: Left: high gradient "edge" regions pointing in the lower half plane. Right: corresponding Hough transform. As labeled here, the angle θ corresponds to the lines' actual orientations instead of the orientation of the normal to the lines. The grey–scale bar on the far right indicates the number of votes for each line in the Hough transform.

4.2 Separation of bimodal sources

Artificial bimodal sources were constructed from the same audio files used in the previous section. The mass was intentionally not evenly distributed between the two peaks in each source. The mass ratio between the peaks was 5/3 in one source and 5/2 in the other. Since gradients are used, the amplitudes of the peaks are not as essential as they are in density estimation

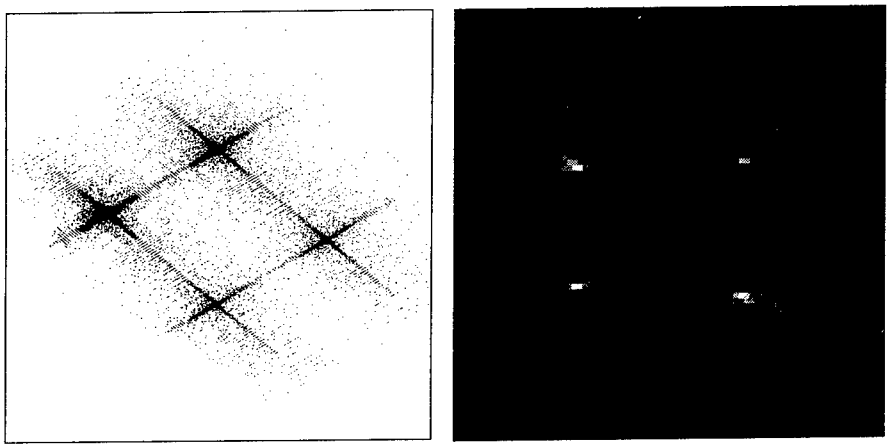


Figure 3: Separation of bimodal sources. Left: input distribution. Right: Hough transform with four peaks corresponding to the four lines. The figure axes labels are the same as that for previous figures.

source separation approaches. The input distribution, large gradient regions, and its Hough transform are shown in Figure 3. Again using the the two most popular line orientations from the Hough transform, the separation achieved a reduction of the mixture to 1 and 1.6 percent of the sources respectively for the two outputs.

4.3 Separation of sources from fewer mixtures

Not only does this approach work for multi–modal sources, the "non–square" blind separation problem where the number of mixtures is less than the number of sources can also be tackled. Extra sources will just contribute extra density extrema directions. The audio sources were pre–processed to make them even more sharply peaked to ease the feature extraction process. Simulation results are shown in Figure 4. The three peaks in the Hough transform now correspond to the three sources in the mixture. The three most popular line orientation peaks found in the Hough transform deviate by only 0.009, 0.01 and 0.002 radians from the actual orientations as determined by the mixing matrix. However, this information is not sufficient to construct the two–to–three dimensional unmixing map. The problem is not well posed since the mixing transformation is not 1-to-1. Nevertheless, a simple "one channel" representation of the three sources can be obtained by partitioning the input space in accordance with the three high density source orientations [7]. A given input $\vec{x}$ is attributed to the source with the closest corresponding orientation, with the value of the output taken to be the dot product of the input vector $\vec{x}$ with a vector of unit length along the source orientation. In this unmixing scheme, only one output is non–zero at any given time, hence

the overlap between any two outputs is zero. Despite the severity of this constraint, the fidelity of the resulting one–channel source approximations is remarkably good, as seen in Figure 5.

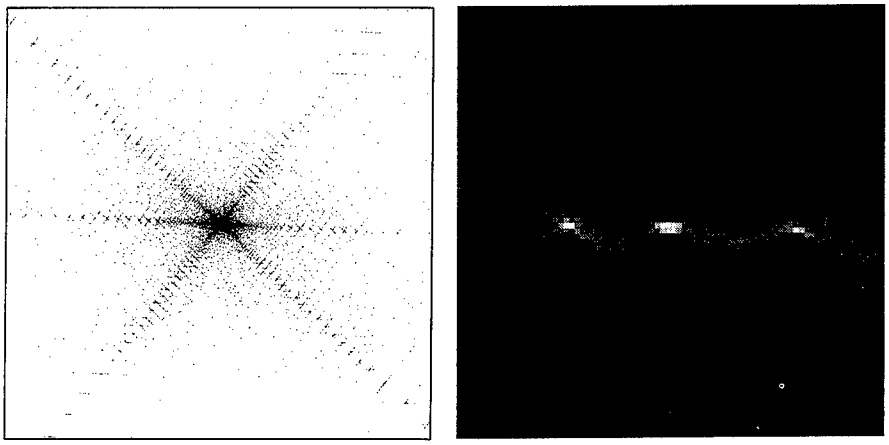


Figure 4: Pseudo–separation of three sharply peaked sources. Left: input distribution. Right: Hough transform with three peaks corresponding to the three sources.

5 CONCLUSION

The algorithm presented in this paper applies specifically to the separation of two dimensional data sets. However, this approach is clearly more general than the specific separation examples presented. The algorithm is applicable to source distributions which contain large derivatives, such as uniform sources. While the Hough transform used is optimized for detecting lines in two dimensions, other versions exist which are sensitive to more general features at a range of spatial scales. It is also clear that the approach introduced here is not limited to two dimensional or linearly mixed signals.

The success of this edge–detection algorithm suggests many possible variations. For example, if the sources are known to be unimodal and sharply peaked, an even simpler algorithm can be used. By normalizing all the input vectors to unit length, the task of finding the N independent component basis vectors becomes that of finding N clusters on the N–sphere. The algorithm presented in this paper works for a larger class of sources because it extracts information from all high gradient regions in the joint distribution. From a mathematical standpoint, we are essentially looking at the iso–probability lines in large gradient areas, with the understanding that they are oriented preferentially along one of the independent component basis directions. A generalization of the algorithm consists of cluster analysis of the local gradient directions.

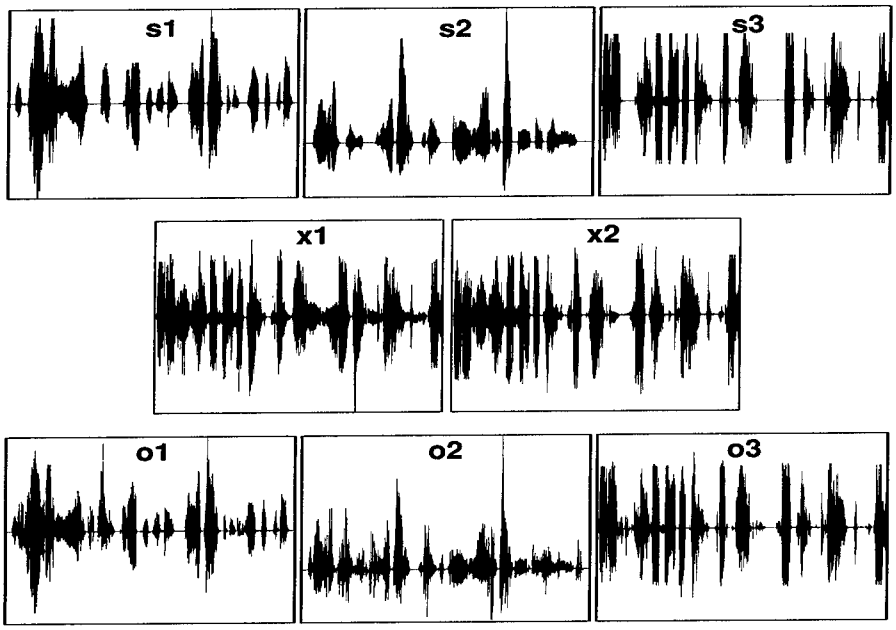


Figure 5: The three sources $(s1, s2, s3)$, two mixtures $(x1, x2)$ and the three outputs $(o1, o2, o3)$ obtained by partitioning the input space into three regions. The outputs were permuted and arbitrarily multiplied by a scaling factor to match the sources. Even though only one output deviates from zero at any given time, the source approximation is surprisingly good. This is seen visually above, and can be verified by listening to the outputs.

The simplicity of this approach should lead to state–of–the–art algorithms in terms of both speed and generality. Not only is the local feature approach more robust to noise than density estimation approaches, the performance degrades gracefully when extra sources are introduced. And finally, from a neural modeling perspective, this approach complements the current work on visual cortex modeling. Bell and Sejnowski [3] found that local edge filters resulted when natural images where fed into their source separation algorithm. In this paper, we show that edge–detectors can perform source separation, and hence the same neural architecture that codes and processes images can also extract independent sources.

References

[1] Amari, S., Cichocki, A., Yang, H. 1995. A new learning algorithm for blind signal separation. *NIPS*8.*

[2] Bell, A. J., and Sejnowski, T. J. 1995. An information-maximization approach to blind separation and blind deconvolution. *Neural Computation.* **7**,1129-1159.

[3] Bell, A. J., and Sejnowski, T. J. 1996. Edges are the "independent components" of natural scenes. *NIPS*9.*

[4] Jahne, B. 1991. **Digital image processing: concepts, algorithms and scientific applications**. Springer-Verlag, Heidelberg.

[5] Cardoso, J-F., and Laheld, B. 1996. Equivariant adaptive source separation. *IEEE Transactions on Signal Processing* **vol. 44**, no 12, pp. 3017-3030.

[6] Lin, J. K., Grier, D. G. and Cowan, J. D. 1996. Source separation and density estimation by faithful equivariant SOM. *NIPS*9.*

[7] Lin, J. K., Grier, D. G. and Cowan, J. D. 1997. Faithful representation of separable distributions. *Neural Computation.* **9**, 1303-1318.

[8] Pearlmutter, B. A. and Parra, L. 1996. Maximum likelihood blind source separation: a context–sensitive generalization of ICA. *NIPS*9.*

BLIND SOURCE SEPARATION OF NONLINEAR MIXING MODELS

Te-Won Lee
Salk Institute, CNL
La Jolla, C.A. 92037, USA
tewon@salk.edu
http://www.cnl.salk.edu/~tewon
Tel.: (619) 453-4100x1215
Fax: (619) 587-0417

Bert-Uwe Koehler
Reinhold Orglmeister
Technische Universitaet Berlin
Institut fuer Elektronik
Einsteinufer 17, 10587 Berlin
koehler@tubife1.ee.tu-berlin.de
orglm@tubife1.ee.tu-berlin.de

Abstract

We present a new set of learning rules for the nonlinear blind source separation problem based on the information maximization criterion. The mixing model is divided into a linear mixing part and a nonlinear transfer channel. The proposed model focuses on a parametric sigmoidal nonlinearity and higher order polynomials. Our simulation results verify the convergence of the proposed algorithms.

1 INTRODUCTION

In blind source separation or independent component analysis (ICA) the problem is how to recover independent sources given the sensor outputs in which the sources have been mixed in an unknown channel. The problem has become increasingly important in the signal processing area due to their prospective application in speech recognition, telecommunications and medical signal processing. The linear blind source separation problem has been studied by researchers in the field of neural networks [1, 2, 5, 9] and statistical signal processing [4, 6]. Potential application in automatic speech recognition systems has been considered in [10] where two speech signals recorded in a real environment have been separated. Furthermore, Makeig et al. [12] have studied independent components of electroencephalographic (EEG) data. There are several other potential applications in the signal processing area which may benefit from ICA as a preprocessing analysis. Nevertheless, the linear mixing model may not be appropriate for some real environment experiments. Therefore, researchers have recently started addressing the ICA problem to nonlinear mixing models [3, 8, 11, 13, 14, 15]. In [8, 11, 13] the nonlinear components are extracted using self-organizing-feature-maps (SOFM). However, due to the limited number of neurons that

0-7803-4256-9/97/$10.00

map the underlying distribution the derived components have a quantization error that increases with increasing distance to the neighboring neurons.

In this paper, we propose a set of algorithms for the nonlinear mixing problem using parametric nonlinear functions. In particular, we assume that the mixing is performed in two stages: a linear mixing followed by a nonlinear transfer function. We focus on a parametric sigmoidal nonlinearity and on higher order polynomials. A similar approach has been independently studied by Taleb and Jutten [14]. They approximate the inverse transfer function by multilayer perceptrons (MLP) that are trained in an unsupervised manner. This kind of model may be justified for several biomedical signal analysis problems such as brain blood moving analysis in magnetic resonance imaging (MRI) and EEG analysis. It may also be used to account for microphone nonlinearities in speech recording experiments. For these problems this model may be an appropriate representation of the actual physical phenomenon.

We present the nonlinear model and derive a set of learning rules based on the information maximization criterion [2]. The learning rules are verified via simulation and future research is discussed at the end.

2 NONLINEAR MIXING AND UNMIXING MODEL

Figure 1 shows the mixing system which is divided into a linear mixing part and a nonlinear transfer part. Each channel i consists of an invertible nonlinear transfer function $f_i(t_i)$. The unmixing system is the inverse sequence of

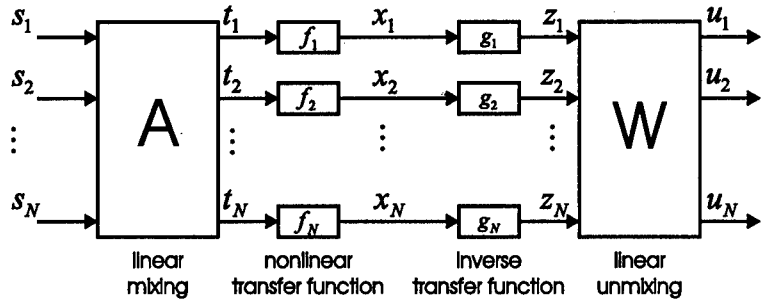


Figure 1: Mixing and Unmixing Model: The mixing stage consists of a linear mixing matrix $\mathbf{A}$ and a nonlinear transfer function $\mathbf{f(t)}$. The unmixing stage consists of the inverse operation - the equalization of the nonlinear transfer function $\mathbf{g(x)}$ and the unmixing matrix $\mathbf{W}$

the mixing system. Figure 1 shows that we first invert the nonlinear transfer function in each channel i with $g_i(x_i)$ and than unmix the linear mixing by applying $\mathbf{W}$ to $\mathbf{z}$. The sources $\mathbf{s}$ are recovered if $g_i(x_i)$ and $\mathbf{W}$ are the inverse functions for $f_i(t_i)$ and $\mathbf{A}$ respectively.

In our model we use the following signals: $\mathbf{s} = [s_1, s_2 \ldots, s_N]^T$, $\mathbf{t} = [t_1, t_2, \ldots, t_N]^T$, $\mathbf{x} = [x_1, x_2, \ldots, x_N]^T$, $\mathbf{z} = [z_1, z_2, \ldots, z_N]^T$,

$\mathbf{u} = [u_1, u_2, \ldots, u_N]^T$, $\mathbf{f} = [f_1(t_1), f_2(t_2), \ldots, f_N(t_N)]^T$, $\mathbf{g} = [g_1(x_1), g_2(x_2), \ldots, g_N(x_N)]^T$. Furthermore, the signals are related by the following equations:

$$\mathbf{t} = \mathbf{A} \cdot \mathbf{s} \tag{1}$$

$$\mathbf{x} = \mathbf{f}(\mathbf{t}) \tag{2}$$

$$\mathbf{z} = \mathbf{g}(\mathbf{x}) \tag{3}$$

$$\mathbf{u} = \mathbf{W} \cdot \mathbf{z} = \mathbf{W} \cdot \mathbf{g}[\mathbf{f}(\mathbf{A} \cdot \mathbf{s})] \tag{4}$$

3 THE LEARNING RULES

3.1 Information Maximization

The separation of independent components from a mixed signal observation $\mathbf{x}$ can be described by a general measure of independence between the pdf of the random variable $p_{\mathbf{x}}(\mathbf{x})$ and the pdf of its components $\prod_{i=1}^{n} p_{x_i}(x_i)$. The Kullback-Leibler divergence measures the degree of distance defined by:

$$\delta(p_{\mathbf{x}}(\mathbf{x}), \prod_{i=1}^{n} p_{x_i}(x_i)) = \int p_{\mathbf{x}}(\mathbf{x}) \log \frac{p_{\mathbf{x}}(\mathbf{x})}{\prod_{i=1}^{n} p_{x_i}(x_i)} d\mathbf{x} \tag{5}$$

and vanishes if and only if $p_{\mathbf{x}}(\mathbf{x})$ factories which leads to $\delta(p_{\mathbf{x}}(\mathbf{x}), \prod_{i=1}^{n} p_{x_i}(x_i)) = 0$. The observation is $\hat{p}_{\mathbf{x}}(\mathbf{x}) = \prod_{i=1}^{n} p_{x_i}(x_i)$ and the Kullback-Leibler divergence have the form of the mutual information of $\mathbf{x}$ and this can be rewritten in terms of entropies as follows:

$$\delta(p_{\mathbf{x}}(\mathbf{x}), \hat{p}_{\mathbf{x}}(\mathbf{x})) = H(p_{\mathbf{x}}(\mathbf{x})) - H(p_{\mathbf{x}}(\mathbf{x}) \mid \hat{p}_{\mathbf{x}}(\mathbf{x})) \tag{6}$$

Bell and Sejnowski [2] have proposed an information-theoretic approach where they maximize the mutual information that an output $y = h(x)$ of a neural processor contains about its input x. They have shown that for invertible and continuous deterministic mappings $h(x)$, the mutual information between inputs and outputs can be maximized by maximizing the entropy of the outputs alone where the output pdf satisfies:

$$p_{\mathbf{y}}(\mathbf{y}) = \frac{p_{\mathbf{x}}(\mathbf{x})}{|J|} \tag{7}$$

with $J(\mathbf{x})$ being the determinant of the Jacobian of the neural transfer function $h(x)$. The Entropy of the Signal y is given by

$$H(\mathbf{y}) = -E[\ln p_{\mathbf{y}}(\mathbf{y})] = E[\ln |J|] - E[\ln p_{\mathbf{x}}(\mathbf{x})] \tag{8}$$

where E denotes the expected value. The maximization is done by maximizing the first term with respect to the parameters of the unmixing functions. That is, we have to learn the elements of the linear unmixing matrix $\mathbf{W}$ and the set of parameters for the nonlinearities $g_i(x_i)$. Using a gradient ascent

algorithm we take the derivative of the entropy function with respect to the w_{ij} and the parameters of the nonlinearity. Therefore, we derive

$$\Delta \mathbf{W} \propto \frac{\partial H(\mathbf{y})}{\partial \mathbf{W}} = \frac{\partial}{\partial \mathbf{W}} \ln |J| \tag{9}$$

The second term in equation (8) is independent of all model parameters. Hence, the gradient of equation (9) is as follows:

$$\frac{\partial}{\partial \mathbf{W}} \ln |J| = \frac{\partial}{\partial \mathbf{W}} \ln |\det(\mathbf{W})| + \frac{\partial}{\partial \mathbf{W}} \ln \left(\prod_{i=1}^{N} \frac{\partial y_i}{\partial u_i} \right) + \frac{\partial}{\partial \mathbf{W}} \ln \left(\prod_{i=1}^{N} \frac{\partial g_i}{\partial x_i} \right) \tag{10}$$

Considering the set of parameters $\mathbf{W}$, a better way to maximize entropy in the feedforward and feedback system is not to follow the entropy gradient, as in [2], but to follow its 'natural' gradient, as reported by Amari et al [1]:

$$\Delta \mathbf{W} \propto \frac{\partial H(\mathbf{y})}{\partial \mathbf{W}} \mathbf{W}^T \mathbf{W} \tag{11}$$

This is an optimal rescaling of the entropy gradient. It simplifies the learning rule and speeds convergence considerably.

3.2 Learning Rules for Sigmoidal Nonlinear Mixing

The infomax criterion holds for our model since independent variables cannot become dependent by passing them through an invertible nonlinearity. Hence, the mutual information before and after the nonlinear stage is not affected.

For the derivation of the learning rule for the w_{ij} we do not need to consider the last term of equation (10). Therefore, the learning rule for $\mathbf{W}$ is:

$$\Delta \mathbf{W} \propto (\mathbf{W}^T)^{-1} + (1 - 2\mathbf{y}) \mathbf{g}^T(\mathbf{x}) \tag{12}$$

considering the Amari et al. extension from equation (11) it follows:

$$\Delta \mathbf{W} \propto \mathbf{W} + (1 - 2\mathbf{y}) \mathbf{u}^T \mathbf{W} \tag{13}$$

Although this learning rule is derived for super-Gaussian sources we may extend the rule to the separation of sub-Gaussian sources by including the kurtosis into the second term which makes the anti-Hebbian rule to a Hebbian learning rule for sub-Gaussians. Girolami and Fyfe use this in the projection pursuit network [7]. In order to derive the complete set of learning rules we assume that the nonlinear mixing is accomplished by a sigmoidal transfer function.

$$f_i(t_i) = \delta_i \left(1 - \frac{2}{1 + exp(-\sigma_i t_i)} \right) \tag{14}$$

where δ denotes the scaling and σ the slope of the transfer function. For this case, $g_i(x_i)$ provides the inverse function by

$$g_i(x_i) = -2r_i \text{arctanh}(d_i x_i) \tag{15}$$

whereas $r_i = 1/\sigma_i$ and $d = 1/\delta_i$ in the ideal case. We perform a gradient ascent on the entropy function to learn the parameters d, r.

$$\Delta \mathbf{r} \propto \frac{\partial H}{\partial \mathbf{r}} \quad \text{and} \quad \Delta \mathbf{d} \propto \frac{\partial H}{\partial \mathbf{d}} \tag{16}$$

$$\frac{\partial H}{\partial \mathbf{r}} = \frac{\partial}{\partial \mathbf{r}} \ln |\det(\mathbf{W})| + \frac{\partial}{\partial \mathbf{r}} \ln \left(\prod_{i=1}^{N} \frac{\partial y_i}{\partial u_i} \right) + \frac{\partial}{\partial \mathbf{r}} \ln \left(\prod_{i=1}^{N} \frac{\partial g_i}{\partial x_i} \right) \tag{17}$$

and

$$\frac{\partial H}{\partial \mathbf{d}} = \frac{\partial}{\partial \mathbf{d}} \ln |\det(\mathbf{W})| + \frac{\partial}{\partial \mathbf{d}} \ln \left(\prod_{i=1}^{N} \frac{\partial y_i}{\partial u_i} \right) + \frac{\partial}{\partial \mathbf{d}} \ln \left(\prod_{i=1}^{N} \frac{\partial g_i}{\partial x_i} \right) \tag{18}$$

The term $\det \mathbf{W}$ in the equations (17) and (18) is independent from $\mathbf{r}$ and $\mathbf{d}$. Hence, we can write:

$$\begin{aligned} \frac{\partial}{\partial r_j} \ln \left(\prod_{i=1}^{N} \frac{\partial y_i}{\partial u_i} \right) &= \sum_{i=1}^{N} (1 - 2y_i) w_{ij} \frac{\partial}{\partial r_j} g_j(x_j) \\ &= -2\operatorname{arctanh}(d_j x_j) \sum_{i=1}^{N} (1 - 2y_i) w_{ij} \end{aligned} \tag{19}$$

and

$$\begin{aligned} \frac{\partial}{\partial d_j} \ln \left(\prod_{i=1}^{N} \frac{\partial y_i}{\partial u_i} \right) &= \sum_{i=1}^{N} (1 - 2y_i) w_{ij} \frac{\partial}{\partial d_j} g_j(x_j) \\ &= -2r_j \frac{x_j}{1 - d_j^2 x_j^2} \sum_{i=1}^{N} (1 - 2y_i) w_{ij} \end{aligned} \tag{20}$$

The third term is then

$$\frac{\partial}{\partial r_j} \ln \left(\prod_{i=1}^{N} \frac{\partial g_i}{\partial x_i} \right) = \frac{1}{r_j} \tag{21}$$

and

$$\frac{\partial}{\partial d_j} \ln \left(\prod_{i=1}^{N} \frac{\partial g_i}{\partial x_i} \right) = 1 + 2d_j^2 x_j^2 (1 - d_j^2 x_j^2)^{-1} \tag{22}$$

3.3 Learning Rules for Flexible Nonlinearities

A weakness of the sigmoidal nonlinearity is that the learning rules can be successfully applied to only those problems which fit to the parametric structure of a sigmoid. However, in certain situations where the a priori knowledge

about the mixing model is not given we need to learn a more flexible nonlinear transfer function. We assume that a nonlinearity may be approximated by polynomials of n-th order. This nonlinear stage my be described as:

$$f_j(t_j) = \sum_{k=1}^{Q} f_{jk} \cdot t_j^{k-1} \tag{23}$$

The inverse $g_j(x_j)$ of the function in equation (23) results in an expression which is generally not easy to handle with respect to our purposes. We therefore make the assumption that the inverse may be approximated by P-th order polynomials. Hence, the inverse is:

$$g_j(x_j) = \sum_{k=1}^{P} g_{jk} \cdot x_j^{k-1} \tag{24}$$

In the same manner, we can perform a gradient ascent on the entropy function to learn the parameters g_{jk}:

$$\Delta g_{jk} \propto \frac{\partial H}{\partial g_{jk}} \tag{25}$$

Performing this operation on equation (8), the learning rule for finding g_{jk} is the sum of the following two terms:

$$\begin{aligned} \frac{\partial}{\partial g_{jk}} \ln \left(\prod_{i=1}^{N} \frac{\partial y_i}{\partial u_i} \right) &= \sum_{i=1}^{N} (1 - 2y_i) w_{ij} \frac{\partial}{\partial g_{jk}} g_j(x_j) \\ &= x_j^{k-1} \sum_{i=1}^{N} (1 - 2y_i) w_{ij} \end{aligned}$$

and

$$\frac{\partial}{\partial g_{jk}} \ln \left(\prod_{i=1}^{N} \frac{\partial g_i}{\partial x_i} \right) = \frac{x_j^{k-2}(k-1)}{\sum_{m=1}^{P} (m-1) \cdot g_{j,m} x_j^{m-2}} \tag{26}$$

4 SIMULATION RESULTS

4.1 Sigmoidal Nonlinearities

To verify the validity of the model and the convergence of the learning rules, we have performed several experiments with the architecture shown in Figure 1. Figure 2 shows the result of the mixing and unmixing system. Two independent white noise sources with super-Gaussian distribution have been generated artificially and are shown in a scatter plot in Figure 2 (a). The sources have been first linearly mixed (b) and then nonlinearly mixed (c).

$$\begin{bmatrix} t_1(z) \\ t_2(z) \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} \tag{27}$$

$$x_1 = f_1(0.5t_1)$$
$$x_2 = f_2(t_2) \tag{28}$$

The unmixing results in Figure 2 (d) and (e) when the nonlinearities are initialized identically and the unmixing matrix $\mathbf{W}$ is chosen randomly. The algorithm converges after presenting 500 samples and the unmixed signals are shown in Figure 2(f). The SNR for the observed mixed signals x_1, x_2 are $-6.3dB$ and $-6.1dB$ respectively. For the unmixed signals u_1, u_2 the SNR is increased to $8.9dB$ and $8.0dB$ respectively.

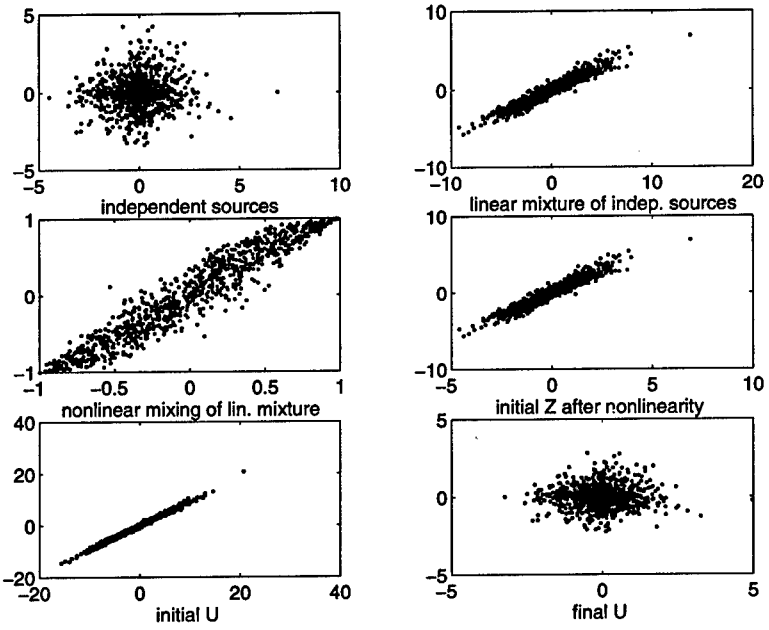


Figure 2: Mixing and Unmixing Simulation. (a) independent sources (b) linear mixed sources (c) nonlinear mixing (d) initially unmixed nonlinearity (e) initial separated signals u (f) final separated signals u

4.2 Flexible Nonlinearities

As in chapter 4.1, we performed several experiments with the architecture shown in Figure 1 to verify the learning rules for flexible nonlinearities. For the linear and nonlinear mixing stage we use the same mixing matrix and nonlinearities as in section 4.1. The independent sources are white noise signals with sub-Gaussian distribution. A scatter plot of the sources is depicted in Figure 3 (a). The unmixing $\mathbf{W}$ and the coefficients g_{jk} of the nonlinearity forming polynomials are chosen randomly with $Q = P$. Figure 3 (f) shows the results after presenting 1000 samples. We observe that the order of the inverse nonlinearity $g(\mathbf{x})$ has to be higher than the order of the nonlinearity

$f(\mathbf{t})$. The stability of the polynomial nonlinearity is highly dependent on the initial value of the coefficients and may be chosen to approximate an invertible nonlinearity.

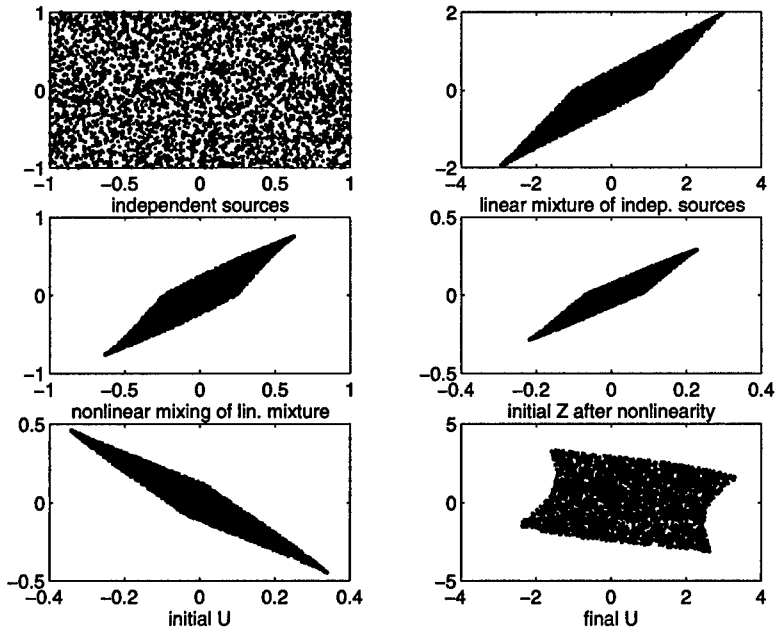


Figure 3: Mixing and Unmixing Simulation Using Flexible Nonlinearities. (a) independent sources (b) linear mixed sources (c) nonlinear mixing (d) initially unmixed nonlinearity (e) initial separated signals u (f) final separated signals u

In figure 4 a) and b) we show the time course signal of a sinusoid and a white noise signal with super-Gaussian distribution. The signals have been mixed linearly and transformed by a nonlinear transfer function $f(t)$ where $f(t)$ is an invertible 5th-order polynomial function. The inverse is approximated by a 8th-order polynomial function $g(x)$. The time course of the recovered signals are shown in figure 4 e) and f).

5 CONCLUSIONS AND FUTURE RESEARCH

We have derived a set of learning rules for the nonlinear blind source separation problem based on the information maximization criterion. The mixing model is divided into a linear mixing part and a nonlinear transfer channel. The proposed algorithms are focused on a parametric sigmoidal nonlinearity and higher order polynomials. Simulation results have been performed to verify the learning rules.

We plan to apply the algorithms to biomedical data such as MRI. To this end, we need to investigate further the stability and convergence criterion of the proposed algorithms. In addition, the model can be extended to exhibit

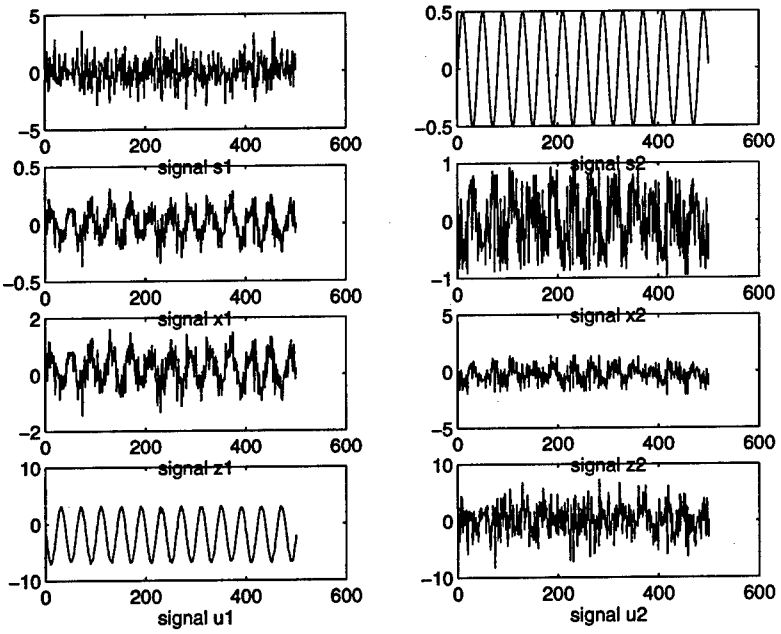


Figure 4: Mixing and Unmixing Simulation Using Flexible Nonlinearities.

a nonlinear cross channel mixing. Then, instead of N nonlinearities for N channels we have to find N^2 nonlinearities. Cross channel nonlinearities have been considered by Yang et al. in [15] and Burel in [3]. In their approach, the nonlinear observation $\mathbf{g(x)}$ has been linearly mixed by a second mixing matrix $\mathbf{W_2}$. The learning rules can be derived to find $\mathbf{W_1}$, $\mathbf{W_2}$ and the nonlinearity $\mathbf{g(x)}$. In contrast to their approach, subject of our future interest is to find nonlinear cross-channels which can be parameterized independently from the channel transfer functions.

ACKNOWLEDGMENTS

T.W.L. is supported by the Daimler-Benz-Fellowship. We are grateful to Tony Bell, Christian Jutten, Anisse Taleb, Scott Makeig and Terry Sejnowski for discussions and comments.

References

[1] S. Amari, A. Cichocki, and H. Yang. A New Learning Algorithm for Blind Signal Separation. In *Advances in Neural Information Processing Systems 8*, 1996.

[2] A.J. Bell and T.J. Sejnowski. An Information-Maximization Approach to Blind Separation and Blind Deconvolution. *Neural Computation*, 7:1129–1159, July 1995.

[3] G. Burel A non-linear neural algorithm. Neural networks, vol.5:937–947, 1992.

[4] J-F. Cardoso and B. Laheld. Equivariant adaptive source separation. IEEE Trans. on S.P., Dec. 1996.

[5] A. Cichocki J. Cao, S. Amari. Blind separation of delayed and convolved signals with self-adaptive learning rate. In *Proc. NOLTA'96*, 1996.

[6] P. Comon. Independent component analysis – a new concept? *Signal Processing*, 36(3):287–314, 1994.

[7] M. Girolami and C. Fyfe. Negentropy and kurtosis as projection pursuit indices provide generalised ica algorithms. In *Advances in Neural Information Processing Systems Workshop 9*, 1996.

[8] M. Herrmann and H. Yang. Perspectives and limitations of self-organizing maps. In ICONIP'96 . In press.

[9] J. Karhunen, E. Oja, L. Wang, R. Vigario, and J. Joutsensalo. A class of neural networks for independent component analysis. IEEE Trans. on Neural Networks, vol8:487–504, May. 1997.

[10] T-W. Lee, A.J. Bell and R. Orglmeister. Blind source separation of real-world signals. *Proc. ICNN*, Houston, USA, 1997.

[11] J.K. Lin and J.D. Cowan. Faithful representation of separable input distributions. *Neural Computation*, to appear, submitted 1996.

[12] S. Makeig, A.J. Bell, T-P. Jung, T.J. Sejnowski. Independent component analysis of electroencephalographic data In *Advances in Neural Information Processing Systems 8*. MIT Press, 1996.

[13] P. Parjunen, A. Hyvaerinen, J. Karhunen. Nonlinear blind source separation by self-organizing maps. In ICONIP'96 . In press.

[14] A. Taleb and C. Jutten. Nonlinear source separation: The post-nonlinear mixtures. In ESANN'97 . In press.

[15] H.H. Yang, S. Amari, A. Cichocki. Information Back-propagation for Blind Separation of Sources from Non-linear Mixtures. In ICNN'97 . Proc. of ICNN'97, Houston.

BLIND SOURCE SEPARATION: ARE INFORMATION MAXIMIZATION AND REDUNDANCY MINIMIZATION DIFFERENT?

D. Obradovic and G. Deco
Siemens AG, Central Technology Department, ZT IK 4
Otto-Hahn-Ring 6, 81739 Munich, Germany

Abstract

This paper provides a detailed and rigorous analysis of the two commonly used methods for blind source separation: Linear Independent Component Analysis (ICA) and Information Maximization (InfoMax). The paper shows analytically that ICA based on the Kullback-Leibler information as a mutual information measure and InfoMax lead to the same solution if the parameterization of the output nonlinear functions in the latter method is sufficiently rich. Furthermore, this work discusses the alternative redundancy measures not based on the Kullback-Leibler information distance and Nonlinear ICA. The practical issues of applying ICA and InfoMax are also discussed.

1. INTRODUCTION

The pioneer work of Zipf [1] and the ideas of Attneave [2] about information processing in visual perception have led to the idea that nervous system and brain may be regulated by an economy principle. In the neural network society these ideas were introduced by the important paper of Barlow [3]. In this work the author presented the connectionist model of unsupervised learning under the perspective of redundancy reduction. The minimum entropy coding method was introduced for the generation of factorial codes [4]. Atick and Redlich [5] demonstrated that statistically salient input features can be optimally extracted from a noisy input by maximizing mutual information. Simultaneously, Atick and Redlich [6] and specially the works of Redlich ([7], [8]) concentrate on the original idea of feature extraction by redundancy reduction. Several neural network learning algorithms for PCA are presented, among others, in [9] and [10].

The problem of Linear Independent Component Analysis as linear feature extraction, i.e. blind source separation was introduced by Comon [11] and further extended in linear and defined in nonlinear case by the works of the authors ([12]-[19]). In parallel, Bell and Sejnowski [20] have demonstrated that their InfoMax method can also achieve linear feature extraction. This paper provides a detailed and rigorous analysis of the two methods and derives conditions under which these

0-7803-4256-9/97/$10.00

methods lead to identical solution. In addition, the paper briefly addresses the cumulant based criteria for ICA as well as Nonlinear ICA.

2. LINEAR INDEPENDENT COMPONENT ANALYSIS AND INFORMATION MAXIMIZATION

Let $\hat{x}$ be random vector of dimension n with the joint probability density function $p(\hat{x})$ whose covariance matrix is nonsingular. Furthermore, let M be a linear square map which maps $\hat{x}$ into the random vector $\hat{y}$ whose probability density function is $p(\hat{y})$.

Definition 1: ICA

Linear Independent Component Analysis (ICA) is an input/output linear transformation M from $\hat{x}$ to $\hat{y}$ such that the output components with joint probability:

$$p(\hat{y}) = p(y_1 \cdots y_n); \quad \hat{y} = M\hat{x} \tag{1}$$

are "as independent as possible" according to the appropriate measure, i.e. distance $\mathcal{D}$:

$$\mathcal{D}\left(p(\hat{y}), \prod_{1}^{n} p(y_i) \right) \tag{2}$$

In the special case where the complete independence of the output components is achieved, the following holds:

$$p(y_1 \cdots y_n) = p(y_1) \cdots p(y_n) \tag{3}$$

If the input vector $\hat{x}$ is jointly Gaussian, ICA is equivalent to the problem of diagonalizing the output covariance matrix Q_y which is the standard PCA problem. In order to guarantee the existence of the solution for the ICA problem, we assume that the input signal $\hat{x}$ was originally obtained by the invertible linear mixture of the statistically independent signals $z_1 \cdots z_n$.

Definition 2: Information maximization

Let the above defined random vector $\hat{x}$ be transmitted through a combination of a matrix M and n nonlinear functions f_i; $i = 1 \div n$ such that the resulting components of the output vector $\hat{w}$ are defined as:

$$w_i = f_i(y_i) \qquad \hat{y} = M\hat{x} \tag{4}$$

Under the assumption that the every nonlinear function f_i is differentiable and that its derivative f_i' satisfies

$$\int_{-\infty}^{\infty} f_i' dy_i = 1 \tag{5}$$

the information maximization problem is defined as maximization of the entropy

$$H(\vec{w}) = -\int d\vec{w}\ p(\vec{w})\ \log(p(\vec{w})) \tag{6}$$

over the elements of matrix M and, possibly, the free parameters in the parameterization of f_i. Typical choices for f_i are single or neural networks with normalized sums of sigmoidal functions.

At first glance ICA and InfoMax problems seem to be substantially different. Nevertheless, it is known that the information maximization leads to the statistical factorization of the output components w_i, i.e. that it essentially performs the same task as ICA [20]. In the remaining part of the paper we give a rigorous proof that these two problems are identical when the Kullback-Leibler information is used as a measure of the statistical independence in ICA and when the derivatives f_i' are capable of approximating output marginal distributions with the infinite precision.

The Kullback-Leibler distance between the joint and the marginal probabilities is defined as:

$$K\{p(\vec{y}), \prod_i p(y_i)\} = \int d\vec{y}\ p(\vec{y})\ \log\left(\frac{p(\vec{y})}{\prod_i p(y_i)}\right) \geq 0 \tag{7}$$

or equivalently:

$$K\{p(\vec{y}), \prod_i p(y_i)\} = \sum_{i=1}^{n} H(y_i) - H(\vec{y}) \tag{8}$$

Equation (8) indicates that the Kullback-Leibler distance is the mutual information between the output components y_i.

The relationship between the input and output joint probabilities of a differentiable map g is equal to:

$$p(\vec{out}) = \frac{p(\vec{in})}{|\det(J)|} \tag{9}$$

where J is the Jacobian matrix of g. Consequently, the relationship between the corresponding entropies is:

$$H(p(\vec{out})) = H(p(\vec{in})) + \int d\vec{in}\ p(\vec{in})\ \log(|\det(J)|) \tag{10}$$

Combining equations (7) and (8) with (9), it follows that:

$$\begin{aligned} K\{p(\hat{y}), \prod_i p(y_i)\} &= \int d\hat{x}\ p(\hat{x})\ \log\left(\frac{p(\hat{x})}{\prod_i p(y_i)\cdot|\det(M)|}\right) \geq 0 \\ &= -H(p(\hat{x})) - \int d\hat{x}\ p(\hat{x})\ \log\left(\prod_i p(y_i)\cdot|\det(M)|\right) \geq 0 \end{aligned} \tag{11}$$

Since the input entropy $H(p(\hat{x}))$ is independent of the input-output transformation, the minimization of $K\{p(\hat{y}), \prod_i p(y_i)\}$ is equivalent to maximization of $\int d\hat{x}\ p(\hat{x})\ \log\left(\prod_i p(y_i)\cdot|\det(M)|\right)$, i.e. to the Maximum Likelihood Expectation (MLE) of $\log\left(\prod_i p(y_i)\cdot|\det(M)|\right)$. In general, the analytical expression for the marginal probabilities $p(y_i)$ are not known, and their estimates $\hat{p}(y_i)$ have to be obtained from the data for every change of the matrix M.

Similarly, in the information maximization problem the output joint entropy $H(\hat{w})$ is equal to:

$$\begin{aligned} H(\hat{w}) &= \int d\hat{y}\ p(\hat{y})\ \log\left(\frac{p(\hat{y})}{\prod_i f_i'(y_i)}\right) \\ &= K\{p(\hat{y}), \prod_i f_i'(y_i)\} \\ &= H(p(\hat{x})) + \int d\hat{x}\ p(\hat{x})\ \log\left(\prod_i f_i'(y_i)\cdot|\det(M)|\right) \geq 0 \end{aligned} \tag{12}$$

or, equivalently, to the MLE of $\log\left(\prod_i f_i'(y_i)\cdot|\det(M)|\right)$.

Hence, the ICA with the Kullback-Leibler information measure and the maximum information transfer as defined in this paper are posed as:

$$\begin{aligned} \text{ICA} \quad &\Rightarrow \quad \min \; K\{p(\hat{y}), \prod_i p(y_i)\} \\ \text{INFOMAX} \quad &\Rightarrow \quad \min \; K\{p(\hat{y}), \prod_i f_i'(y_i)\} \end{aligned} \tag{13}$$

or, equivalently, to the following MLE over the input probability $p(\hat{x})$:

$$\begin{aligned} \text{ICA} \quad &\Rightarrow \quad \text{MLE}\{\log\Big(\prod_i \hat{p}(y_i) \cdot |\det(M)|\Big)\} \\ \text{INFOMAX} \quad &\Rightarrow \quad \text{MLE}\{\log\Big(\prod_i f_i'(y_i) \cdot |\det(M)|\Big)\} \end{aligned} \tag{14}$$

A MLE formulation of the InfoMax method is also discussed in [21]-[23]. The initial parameterization of the derivatives $f_i'(y_i)$ in (14) has a possible interpretation as the prior on the estimation of the actual marginal densities $p(y_i)$. As mentioned earlier, both methods require parameterization of $\hat{p}(y_i)$ and $f_i'(y_i)$. Hence, the problem statements in (13) and (14) can be used to derive conditions for the equivalence of solutions of ICA and InfoMax.

Lemma:

For a given input distribution $p(\hat{x})$, the ICA and InfoMax problems achieve the same degree of statistical independence if the derivatives $f_i'(y_i)$ can be parameterized in the form of the marginal distribution estimates $\hat{p}(y_i)$.

The proof is straightforward since it requires that the parameterization of $p(\hat{x})$ and $f_i'(y_i)$ are identical. This can be illustrated on an example.

Example:

The marginal probabilities $p(y_i)$ have to be estimated from the data. A typical way of doing that is to estimate elements of a probability density function expansion up to the desired order. Let us use the first element of the Edgeworth expansion [19], i.e. let $\hat{p}(y_i)$ have the form of a Gaussian whose mean and standard deviation σ_i is equal to the those of the actual marginal distribution $p(y_i)$. Without a loss of generality let us assume that the input distribution $p(\hat{x})$ is zero-mean. In addition, let us parameterize the derivatives $f_i'(y_i)$ as zero-mean Gaussian distributions whose standard deviations r_i are optimization parameters. Hence, it is easy to see that the MLE problems in (14) become

$$\text{ICA} \quad \Rightarrow \quad \text{MLE}\{-\sum[\log(\sigma_i)] + \log(|\det(M)|)\} \qquad \sigma_i = \langle y_i^2 \rangle$$

$$\text{INFOMAX} \quad \Rightarrow \quad \text{MLE}\left\{-\sum\left[\log(r_i) + \frac{y_i^2}{2r_i^2}\right] + \log(|\det(M)|)\right\} \tag{15}$$

The resulting ICA problem is nothing more than the covariance matrix diagonalization [19] where the optimization is performed over the elements of the matrix M. In the case of InfoMax, the unknown parameters are not only the elements of M but also the Gaussian parameters r_i. It is easy to see that the optimal value of r_i for every fixed matrix M is the actual standard deviation σ_i and, therefore, that the solution of InfoMax problem will also result in the covariance matrix diagonalization.

In practice, it is required that the solutions of both methods are unique modulo transformations that preserve statistical independence such as the component order permutation and diagonal scaling. The uniqueness is achieved if the number of Gaussian components of $p(\hat{x})$ does not exceed one. In the case of multiple Gaussian distributions, it is well known that there is an infinite number of matrix transformations that diagonalize the covariance matrix. Hence, the ICA and InfoMax algorithms blind source separation will have unique solutions only if the original signal $\hat{x}$ did not have more than one Gaussian components. In addition, there can be problems concerning the scaling of the elements of the matrix M. Hence, it is the experience of the authors that imposing the condition

$$\det(M) = 1 \tag{16}$$

makes the optimization numerically stable and avoids possible scaling problems. Different parameterizations of M such that the condition in (16) holds can be found in [19].

3. ALTERNATIVE REDUNDANCY MEASURES AND NONLINEAR ICA

The previous section has demonstrated that ICA and InfoMax are identical when the redundancy measure in ICA is the Kullback-Leibler information distance and when sufficient freedom is given to the marginal output probability modeling and estimation. Nevertheless, there are other measures that are easy to implement, especially in the case of a linear mixing with a matrix M. The following part of the paper briefly reviews ICA based on the properties of cumulant expansion of the joint probability density function $p(\hat{y})$. The detailed derivation and analysis of the cumulant based ICA can be found in [17] and [19].

The cumulant based criterion for ICA is derived by comparison of the cumulant expansion of the joint probability density $p(\hat{y})$ and of the product of the marginal output probabilities $p(y_i)$. The complete factorization is achieved if the both expansions are the same, i.e. if the non-diagonal coefficients in the higher order

cumulants of $p(\hat{y})$ take desired values (usually zero) imposed by the statistical independence of $p(y_i)$. Since the cumulant expansion of an arbitrary distribution has infinite number of elements, for practical purposes only cumulants up to the order four are considered. Hence, the resulting ICA cumulants based criterion has the following form:

$$J(M) = \sum_{i=1}^{4} \sum_{nondiag} \left[C^{(i)}_{nondiag} - C^{(i)}_{nondiag-desired} \right]^2 \qquad (17)$$

where i defines the cumulant order, and where $C^{(i)}_{nondiag}$ and $C^{(i)}_{nondiag-desired}$ are the non-diagonal cumulant coefficients and their desired values for a given cumulant order i of the joint probability density function $p(\hat{y})$. In general, the desired coefficients $C^{(i)}_{nondiag-desired}$ are equal to zero. For every change of the matrix M, the non-diagonal coefficients are estimated and the cost function J (M) further minimized.

The cumulant based ICA criterion can be further simplified by using the properties of cumulant expansion when M is a rotation matrix R [17]. In the case of the rotation matrix, the minimization of the criterion in (17), becomes equivalent to maximization of the sum of squared diagonal elements.

$$\min \{J(R)\} = \max \left\{ \sum_{i=1}^{4} \sum_{diag} \left[C^{(i)}_{diag}(R) \right]^2 \right\} \qquad (18)$$

Consequently, the original cumulant cost function is significantly reduced when the linear transformation M is restricted to the set of rotation matrices. Although in general M is not a rotation matrix, we can still try to take advantage of the property described in (18).

The statistical independence implies diagonal structure of the output covariance matrix. Hence, let N be an invertible matrix which diagonalizes the covariance matrix of the input variable $\hat{x}$:

$$N \cdot Q_x \cdot N^T = D_i \qquad (19)$$

where D_i is a nonsingular diagonal matrix according to our assumptions about the original signals and the mixing process. Then, all linear input-output transformations M which result in statistical independence of output components at higher order while preserving the diagonal structure of the covariance matrix of $\hat{y} = M \cdot \hat{x}$ can be parameterized as follows [17]:

$$M = P \cdot D \cdot R \cdot D_i^{-0.5} \cdot N \qquad (20)$$

where P is a permutation matrix, D is an invertible diagonal scaling matrix, and R is a rotation.

Since statistical independence is invariant to the diagonal scaling and permutation, the only free variable after the scaling with $D_1^{-0.5} \cdot N$ is the rotation matrix R. The suitable parameterization of rotation matrices is defined [19]:

$$R = (I+A)^{-1}(I-A); \qquad A^T = -A \tag{21}$$

where A is the skew-symmetric matrix whose number of independent parameters is equal to $0.5n(n-1)$. This parameterization covers all rotation matrices R with the property that $(I+R)$ is nonsingular, i.e. that no eigenvalue of R is equal to -1. The latter condition represents no restriction in our case since there always exists a diagonal unitary matrix which can make the eigenvalues different from -1.

Now, the cost function in (17) can be posed as the optimization problem w.r.t. the matrix R, i.e. A:

$$\max \sum_{i=1}^{4} \sum_{\text{diag}} \left[C_{\text{diag}}^{(i)}(\hat{y}) \right]^2 \tag{22}$$

$$R^T = R^{-1} = (I+A)^{-1}(I-A) \quad \text{and} \quad \hat{y} = R \cdot D_1^{-0.5} \cdot N \cdot \hat{x}$$

The $0.5n(n-1)$ free parameters of the matrix A can be determined by any gradient based method such as backpropagation. Furthermore, the diagonal elements of the cumulant tensors are the cumulants of the individual elements of $\hat{y}$ which, on the other hand, are polynomial functions of the cross-cumulant elements of $\hat{x}$. Consequently, the optimization can be significantly simplified by pre-calculated the cumulants of $\hat{x}$ up to the other i. It is the experience of the authors that the cumulant based ICA criterion in (22) is numerically superior to the Kullback-Leibler distance based ICA. Several applications of the cumulant based blind source separation method presented in (22), including the "Cocktail Party" example and the mixture of the uniform distributed signals, can be found in [19].

As the last point of this section, the authors would like to mention that the ICA problem can be formulated also in the case where the input-output map is not a matrix but an invertible nonlinear function F. A parameterization of such functions with the so called "triangular volume preserving network" is presented in [12] and [19]. The reference [19] presents several applications of the Nonlinear ICA.

4. CONCLUSIONS

This paper has provided a detailed and rigorous analysis of the two commonly used methods for blind source separation: Linear Independent Component Analysis (ICA) and Information Maximization (InfoMax). The paper showed analytically that ICA based on the Kullback-Leibler information as a mutual information measure and InfoMax lead to the same solution if the parameterization of the output nonlinear functions in the latter method is sufficiently rich. Furthermore, this work has discussed the alternative, cumulant based blind source separation

methods and Nonlinear ICA. The practical issues of applying ICA and InfoMax were also discussed.

5. REFERENCES

[1] G. Zipf: Human Behavior and the Principle of Least Effort. Addison-Wesley, Cambridge, Massachusetts, 1949.

[2] F. Attneave: Informational Aspects of Visual Perception. Psychological Review, **61**, 183-193, 1954.

[3] H. Barlow: Unsupervised Learning. Neural Computation, **1**, 295-311, 1989.

[4] H. Barlow, T. Kaushal and G. Mitchison: Finding Minimum Entropy Codes. Neural Computation, **1**, 412-423, 1989.

[5] J. Atick and A. Redlich: Towards a theory of early visual processing. Neural Computation, **2**, 308-320, 1990.

[6] J. Atick and A. Redlich: What Does the Retina Know about Natural Scenes. Neural Computation, **4**, 196-210, 1992.

[7] A.N. Redlich: Redundancy Reduction as a Strategy for Unsupervised Learning. Neural Computation, 5, 289-304, 1993.

[8] A.N. Redlich: Supervised Factorial Learning. Neural Computation, **5**, 750-766, 1993.

[9] G. Deco and D. Obradovic, Principal Component Analysis: A Factorial Learning Approach. International Conference on Artificial Neural Networks, Springer Verlag, vol. 2, 1059-1062, Sorrento, Italy, 1994.

[10] D. Obradovic and G. Deco: Linear Feature Extraction in Networks with Lateral Connections. IEEE World Congress on Computational Intelligence, vol. 2, 686-691, Florida, USA, 1994.

[11] P. Comon: Independent Component Analysis, A new concept? Signal Processing, **36**, 287-314, 1994.

[12] G. Deco and W. Brauer: Higher Order Statistics with Neural Networks. Advances in Neural Information Processing 7, Eds.: G. Tesauro, D. Touretzky and T. Leen, MIT Press (Cambridge) 247-54, 1994.

[13] G. Deco and W. Brauer: Nonlinear Higher Order Statistical Decorrelation by Volume-Conserving Neural Networks. Neural Networks, 8, 525-535, 1995.

[14] G. Deco and B. Schürmann: Learning Time Series Evolution by Unsupervised Extraction of Correlations. Physical Review E, 51, 1780-1790, 1995.

[15] L. Parra, G. Deco and S. Miesbach: Redundancy Reduction with Information Preserving Nonlinear Maps. Networks: Computation in Neural Systems, 6, 61-72, 1995.

[16] G. Deco and D. Obradovic: Rotation Based Redundancy Reduction Learning. Neural Networks, 8, 751-755, 1995.

[17] D. Obradovic and G. Deco: Linear Feature Extraction in non-Gaussian Networks. World Congress on Neural Networks, vol2, 523-527, Washington DC, 1995.

[18] D. Obradovic and G. Deco: An information theory based learning paradigm for linear feature extraction. Neurocomputing, 12, 203-221, 1996.

[19] G. Deco and D. Obradovic: An Information-Theoretic Approach to Neural Computing. SPRINGER VERLAG (New York), February 1996.

[20] A.J. Bell and T.J. Sejnowski: An Information-Maximization Approach to Blind Separation and Blind Deconvolution.Neural Computation, 7, Number 6, 1129-1160, November 1995.

[21] Pearlmutter B.A. and Parra L.C.:. A context-sensitive generalization of ICA. Proc. ICONIP'96, Japan, 1996.

[22] Jean-François Cardoso: Infomax and maximum likelihood for source separation. To appear in IEEE Letters on Signal Processing, April, 1997.

[23] Pham, D.T., Garat, P., Jutten, C.: Separation of a mixture of independent sources through a maximum likelihood approach. Proceedings of EUSIPCO-92, Belgium, 24-27 Aug. 1992.

BLIND SIGNAL DECONVOLUTION BY SPATIO-TEMPORAL DECORRELATION AND DEMIXING

Seungjin CHOI and Andrzej CICHOCKI

Lab for Artificial Brain Systems
Frontier Research Program, RIKEN
2-1 Hirosawa, Wako-shi
Saitama 351-01, Japan
Tel: +81-48-462-1111 ext. 6714
Fax: +81-48-462-4633
E-mail: schoi@lion.riken.go.jp, cia@hare.riken.go.jp

Abstract

In this paper we present a simple efficient local unsupervised learning algorithm for on-line adaptive multichannel blind deconvolution and separation of i.i.d. sources. Under mild conditions, there exits a stable inverse system so that the source signals can be exactly recovered from their convolutive mixtures. Based on the existence of the inverse filter, we construct a two-stage neural network which consists of blind equalization and source separation. In blind equalization stage, we employ anti-Hebbian learning in temporal domain for decorrelation. For blind separation, we can apply any existing algorithms. Extensive computer simulations confirm the validity and high performance of our proposed learning algorithm.

1 INTRODUCTION

Blind signal separation from convolutive mixtures of unknown source signals is a really challenging and fundamental problem encountered in many applications such as cocktail party problem, wideband array signal processing, image processing, digital communication, and some biomedical applications. In real world applications, the observed signals obtained from sensors are usually convolutive mixtures due to the propagating source signals through the dynamic medium and parasitic effects like multiple echoes and reverberation. In this paper, we present a new approach to multichannel blind deconvolution and separation of i.i.d. source signals. In multichannel deconvolution and separation, an m dimensional vector of received signals $\mathbf{x}(k)$ is assumed to be generated from an n dimensional vector of independent source signals $\mathbf{s}(k)$ using the multi-variate linear time invariant filters, i.e.,

$$\begin{aligned} \mathbf{x}(k) &= \sum_{i=0}^{M} \mathbf{H}_i \mathbf{s}(k-i) + \mathbf{n}(k), \\ &= [\mathbf{H}(z)]\mathbf{s}(k) + \mathbf{n}(k), \end{aligned} \tag{1}$$

0-7803-4256-9/97/$10.00

where $\mathbf{H}(z) = \sum_{i=0}^{M} \mathbf{H}_i z^{-i}$ ($\mathbf{H}(z)$ is $m \times n$ polynomial matrix and z^{-i} is delay operator such that $z^{-i}\mathbf{s}(k) = \mathbf{s}(k-i)$) is the channel transfer function and $\mathbf{n}(k)$ is an additive white Gaussian noise. We assume that the number of sensors, m is strictly greater than the number of sources, n. The problem of multichannel deconvolution and separation is to recover the source signals $\mathbf{s}(k)$ from the received signals $\mathbf{x}(k)$, up to scale factor, permutation ambiguity, and an arbitrary delay, i.e., $\hat{\mathbf{s}}(k) = \mathbf{P}\mathbf{\Lambda}\mathbf{D}(z)\mathbf{s}(k)$, where $\mathbf{P}$ is a permutation matrix, $\mathbf{\Lambda}$ is a nonsingular diagonal matrix, and $\mathbf{D}(z) = diag\{z^{-d_1}, \cdots, z^{-d_n}\}$.

Most existing methods was devoted to recover the spatially independent source signals from their convolutive mixtures. The output decorrelation approach was proposed by Compernolle and Gerven [11] and further investigated by [25, 5, 16]. The methods developed in the separation of instantaneous mixtures have been extended to multi-channel deconvolution problem (see [15, 24, 10, 2]). Bussgang approach in frequency domain has been employed by [17, 18, 19]. Most of time-domain method failed to recover the source signals when the channels are nonminimum phase systems. To overcome this difficulty, frequency-domain approach are suggested [17, 18, 19]. However, the frequency-domain approach is block-adaptive because it requires to compute FFT. Thus, time-domain approach is better if we can handle the nonminimum phase channels properly. In most of aforementioned methods, the number of sensors are assumed to be equal to the number of sources and was usually restricted to the case of only two source signals. As will be shown in this paper, a stable exact inverse system for the convolutive mixture model (1) does not exist if there are equal number of sources and sensors, except for trivial cases.

In this paper, we investigate the separation of i.i.d. source signals (precisely speaking, spatially independent and temporally uncorrelated source signals) and develop an on-line adaptive algorithm. Signal separation task is split into two stages: spatio-temporal decorrelation and blind source separation of instantaneous mixtures. As will be shown in this paper, blind equalization (based on second-order statistics only) is able to deconvolve the Multi Input Multi Output (MIMO) FIR channels up to linear mixtures of source signals. Source separation is employed to separate the instantaneous mixtures.

2 THEORETICAL FUNDAMENTALS, BASIC ASSUMPTIONS

Throughout this paper, the following model assumptions are made:

A1) The source $\mathbf{s}(k)$ is zero-mean with non-zero variance, temporally uncorrelated, and spatially independent, i.e.,

$$E\{s_i(k)\} = 0, \forall i, \tag{2}$$

$$E\{s_i^2(k)\} \neq 0, \forall i, k, \tag{3}$$

$$E\{s_i(k)s_i(k-\tau)\} = 0, \forall \tau \neq 0, \tag{4}$$

$$E\{s_i(k)s_j(k-\tau)\} = 0, \forall \tau, i \neq j, \tag{5}$$

where E denotes the expectation operator.

A2) The number of sensors, m is strictly greater than the number of sources n, i.e., $m > n$.

In blind equalization of Single Input Multi Output (SIMO) FIR channels, it has been shown that under mild conditions (for example, channels do not have common zeros), temporal decorrelation of the composite output (sum of the output of equalizer-bank) can equalize the channels. (see [20, 6, 7] for details). This is extended to blind equalization of MIMO FIR channels. In this section, we present fundamental theoretical results: (1) Under what conditions, a stable inverse system (equalizer) for MIMO FIR channels exists?; (2) if there exists a equalizer for MIMO FIR channels, how can we equalize channels? Let $\mathbf{W}(z)$ be the equalizer of MIMO FIR channel $\mathbf{H}(z)$ ($m \times n$ polynomial matrix). For a given channel $\mathbf{H}(z)$, a zero-forcing condition for blind equalization of MIMO channels is given by

$$\mathbf{W}(z)\mathbf{H}(z) = \mathbf{P}\mathbf{\Lambda}\mathbf{D}(z). \tag{6}$$

The transfer function of the channel, $\mathbf{H}(z)$ is an $m \times n$ polynomial matrix having $\binom{m}{n}$ distinct $n \times n$ submatrices. Let $\Delta_i(z)$, $i = 1, 2, \cdots, \binom{m}{n}$, denote the determinants of these submatrices. Let GCD denote the greatest common divisor. In terms of these quantities, the existence of FIR equalizer $\mathbf{W}(z)$ is given by Massey and Sain [22].

Theorem 1 *An FIR inverse system exists, if and only if*

$$GCD[\Delta_1(z), \Delta_2(z), \cdots, \Delta_{\binom{m}{n}}(z)] = z^{-d}, \tag{7}$$

for some $d \geq 0$.

In particular, such FIR equalizer does not exit if $\Delta_i(z)$ have common zeros (except common zeros at origin). Note that this condition can not be satisfied for the case where we have equal number of sensors and sources, except for trivial cases. Let $\mathbf{G}(z)$ be the global system described as $\mathbf{G}(z) = \mathbf{W}(z)\mathbf{H}(z)$. Then $\mathbf{G}(z)$ is also FIR, of the form

$$\mathbf{G}(z) = \sum_{i=0}^{P} \mathbf{G}_i z^{-i}, \tag{8}$$

where P is the upper bound of the order of $\mathbf{G}(z)$. We generalize the zero-forcing condition (6) as

$$\mathbf{W}(z)\mathbf{H}(z) = \mathbf{\Gamma}\mathbf{D}(z) \tag{9}$$

where $\mathbf{\Gamma}$ is an $m \times m$ nonsingular matrix. This generalized zero-forcing condition (9) has been recently investigated by R. Liu [21] when $\mathbf{D}(z) = \mathbf{I}z^{-d}$.

Theorem 2 *Let the channel* $\mathbf{H}(z)$ *satisfy the condition (7). Suppose that the assumptions (A1) and (A2) are satisfied. Then the generalized zero-forcing condition (9) is satisfied if* $\mathbf{y}(k)$ *satisfies*

$$E\{\mathbf{y}(k)\mathbf{y}^T(l)\} = \mathbf{\Gamma}\delta_{kl}, \tag{10}$$

where δ_{kl} *is Kronecker delta equal 1 for k=l, otherwise 0.*

Sketch of proof: Both $\mathbf{H}(z)$ and $\mathbf{W}(z)$ are FIR, so $\mathbf{G}(z) = \mathbf{W}(z)\mathbf{H}(z)$ is also FIR. Then $\mathbf{y}(k) = \sum_{i=0}^{P} \mathbf{G}_i\mathbf{s}(k-i)$. It is easy to see that if $\mathbf{y}(k)$ satisfies (10), then the generalized zero-forcing condition (9) is satisfied.

Thus we can equalize the MIMO channel $\mathbf{H}(z)$ by spatio-temporal decorrelation of the output $\mathbf{y}(k)$ up to linear mixtures. Note that this result is consistent with some results in [12, 13]. Suppose that $\mathbf{H}(z)$ is full rank for all z. This implies that $\mathbf{H}(z)$ is a minimum phase system. Then the source signals $\mathbf{s}(k)$ can be viewed as the normalized innovation sequence of the observed sequence $\mathbf{x}(k)$. One can recover the innovation sequence from $\mathbf{x}(k)$ up to an orthogonal matrix $\mathbf{Q}$. This is emphasized in [12, 13] where whitening is done by linear prediction. In contrast to [12, 13], our approach is more robust in the sense that we do not need exact knowledge of the order of channels.

3 NEURAL NETWORK MODEL AND THE LEARNING ALGORITHMS

Let $\mathbf{W}(z)$ be a generalized zero-forcing equalizer and $\mathbf{U} \in \mathbb{R}^{n\times m}$ be a demixing memoryless network. Provided that the channel $\mathbf{H}(z)$ satisfies the condition (7), then $\mathbf{UW}(z)$ is a stable inverse system of the FIR channel $\mathbf{H}(z)$, and the golbal system $\mathbf{G}(z)$ should satisfy the relation

$$\mathbf{G}(z) = \mathbf{UW}(z)\mathbf{H}(z) = \mathbf{P\Lambda D}(z). \tag{11}$$

Our approach to multichannel blind deconvolution is illustrated in Figure 1.

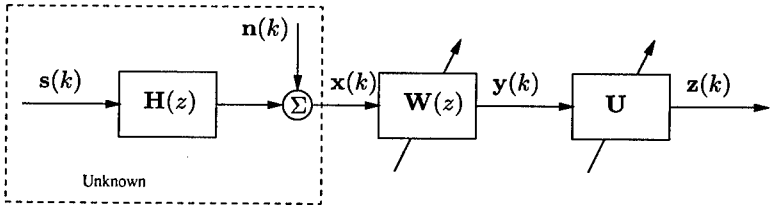


Figure 1: Neural network block diagram for multichannel blind deconvolution: $\mathbf{W}(z)$ represents a generalized zero-forcing recurrent equalizer and matrix $\mathbf{U}$ describes a demixing feedforward memoryless network.

For a generalized zero-forcing equalizer $\mathbf{W}(z)$, we consider a linear feedback network shown in Figure 2. This network is almost fully connected in spatial and temporal domain. Note that it does not have algebraic loops (the connections between $y_i(k)$ and $y_j(k)$ and self-connections from $y_i(k)$ to $y_i(k)$). T he generalized zero-forcing equalizer can be viewed as a whitening filter or spatio-temporal decorrelation filter. The output $y_i(k)$ is described by (see Figure 2)

$$y_i(k) = x_i(k) + \sum_{j=1}^{m}\sum_{p=1}^{L} w_{ijp}(k) y_j(k-p), \text{ for } i = 1, ..., m, \tag{12}$$

where the synaptic weight $w_{ijp}(k)$ is the connection strength between $y_i(k)$ and $y_j(k-p)$. Or in matrix form

$$\mathbf{y}(k) = \mathbf{x}(k) + \sum_{p=1}^{L} \mathbf{W}_p(k)\mathbf{y}(k-p), \tag{13}$$

where $\mathbf{y}(k) = [y_1(k), \cdots, y_m(k)]^T$, $\mathbf{x}(k) = [x_1(k), \cdots, x_m(k)]^T$, and $\mathbf{W}_p(k) = [w_{ijp}]_{m\times m}$ is synaptic weight matrix. From (13), it is easy to find that the equalizer $\mathbf{W}(z)$ can be expressed as

$$\mathbf{W}(z) = (\mathbf{I} - \sum_{p=1}^{L} \mathbf{W}_p z^{-p})^{-1}. \tag{14}$$

For spatio-temporal decorrelation of the output signals $y(k)$, we apply a simple learning algorithm which is a temporal variant of anti-Hebbian learning. The weight matrices $\mathbf{W}_p(k)$ for $p = 1, \cdots, L$ are updated by the following learning algorithm:

$$\mathbf{W}_p(k+1) \quad = \quad \mathbf{W}_p(k) - \eta(k)\{\mathbf{y}(k)\mathbf{y}^T(k-p)\}, \tag{15}$$

where $\eta(k) > 0$ is a learning rate. It is easy to see that the above algorithm (15) achieves the convergence for

$$E\{\mathbf{y}(k)\mathbf{y}^T(k-p)\} = \mathbf{0}, \text{ for } p = 1, \cdots, L. \tag{16}$$

Thus at steady state $\mathbf{y}(k) = \mathbf{\Gamma D}(z)\mathbf{s}(k)$. Note that all stable equilibria of (16) are desirable solution where the output $\mathbf{y}(k)$ are uncorrelated in spatio-temporal domain. Note that the learning algorithm (15) for the generalized zero-forcing equalizer is very simple, local, and biologically plausible. After spatio-temporal decorrelation by a generalized zero-forcing equalizer, the output $\mathbf{y}(k)$ consists of instantaneous mixtures of the source signals. At second stage, source separation feedforward memoryless network is implemented, described as

$$\mathbf{z}(k) = \mathbf{U}(k)\mathbf{y}(k). \tag{17}$$

For the separation of instantaneous mixtures, any existing methods [14, 9, 3, 1, 4, 8] can be employed. In this paper, we employ the following source separation algorithm [1]:

$$\mathbf{U}(k+1) = \mathbf{U}(k) + \eta(k)\{\mathbf{I} - \mathbf{f}(\mathbf{z}(k))\mathbf{z}^T(k)\}\mathbf{U}(k), \tag{18}$$

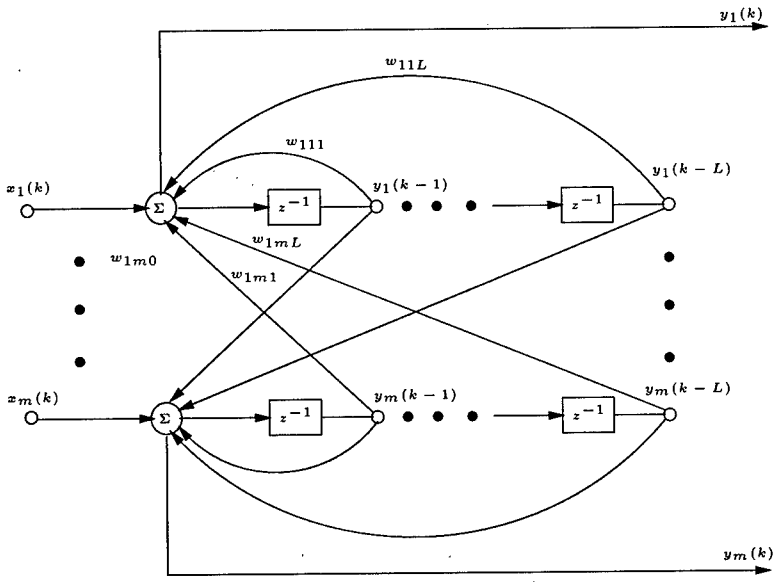


Figure 2: The proposed linear temporal feedback network for the generalized zero-forcing equalizer

where $\mathbf{f}(\mathbf{z}(k)) = [f(z_1(k)), \cdots, f(z_n(k)]^T$ is properly chosen nonlinear function. The choice of this function depends on the statistics of source signals. For example, it is known that $f(z_i(k)) = \beta z_i(k) + z_i^3(k)$ is for sub-Gaussian source signals and $f(z_i(k)) = \beta z_i(k) + \tanh(\alpha z_i(k))$ is for super-Gaussian source signals for some $\beta > 0$.

4 COMPUTER SIMULATIONS

One exemplary simulation result is presented here. In this simulation, three i.i.d. sources and five sensors were used. Three i.i.d. sources consist of random variables that are uniformly distributed over the binary set $\{-1, +1\}$. Five received signals $\mathbf{x}(k)$ were generated by

$$\begin{aligned}\mathbf{x}(k) = \mathbf{H}_0\mathbf{s}(k) + \mathbf{H}_1\mathbf{s}(k-1) + \mathbf{H}_2\mathbf{s}(k-2) \\ +\mathbf{H}_5\mathbf{s}(k-5) + \mathbf{H}_{10}\mathbf{s}(k-10).\end{aligned} \tag{19}$$

The weighting coefficients in mixing/convolutive model , $\mathbf{H}_i$, $i = 0, 1, 2, 5, 10$ were generated randomly over the interval $[-1, 1]$. They were assumed to be com-

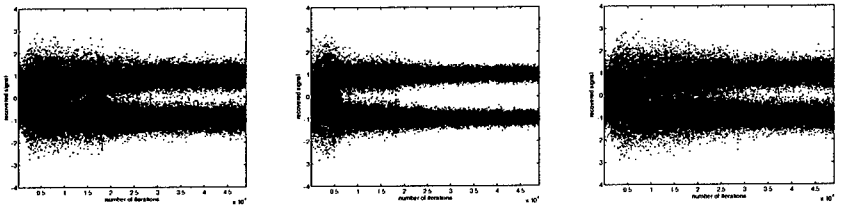
pletely unknown.

$$\mathbf{H}_0 = \begin{bmatrix} -0.8244 & -0.5281 & 0.2690 \\ 0.2790 & -0.4718 & 0.9545 \\ -0.8027 & 0.2088 & 0.6510 \\ 0.3813 & -0.1638 & -0.4125 \\ -0.3169 & -0.7274 & -0.2183 \end{bmatrix}, \mathbf{H}_1 = \begin{bmatrix} -0.7599 & -0.2378 & 0.8745 \\ 0.7246 & -0.2266 & -0.2742 \\ 0.0496 & -0.3765 & 0.3459 \\ 0.9096 & 0.1851 & 0.7272 \\ -0.0720 & -0.2393 & 0.7922 \end{bmatrix},$$

$$\mathbf{H}_2 = \begin{bmatrix} 0.3430 & -0.0721 & 0.3401 \\ 0.0393 & 0.8915 & -0.7729 \\ 0.3706 & -0.0686 & -0.0818 \\ -0.9449 & -0.9798 & -0.7116 \\ -0.8525 & 0.4407 & -0.3565 \end{bmatrix}, \mathbf{H}_5 = \begin{bmatrix} 0.3381 & -0.5663 & 0.1081 \\ 0.9698 & -0.2780 & 0.1904 \\ -0.0709 & 0.1803 & 0.3893 \\ -0.1861 & -0.5072 & -0.9113 \\ 0.8387 & -0.7378 & 0.3675 \end{bmatrix},$$

$$\mathbf{H}_{10} = \begin{bmatrix} -0.1154 & -0.0173 & 0.6804 \\ 0.4578 & 0.4206 & -0.6061 \\ -0.3082 & -0.3343 & -0.1155 \\ 0.1367 & -0.2402 & 0.2099 \\ -0.9465 & 0.9707 & 0.1975 \end{bmatrix}.$$

The eye pattern is shown in Figure 3. Figure 4 shows the original source signals $\mathbf{s}(k)$, convolutive mixtures $\mathbf{x}(k)$, and recovered signals $\mathbf{z}(k)$. Only 51 samples for each are plotted. It can be observed that $z_1(k) = -s_2(k)$, $z_2(k) = s_1(k)$, and $z_3(k) = s_3(k)$. From Figure 4, we can see that the recovery of i.i.d. source signals from their convolutive mixtures are perfect even for case of nonminimum phase channels.

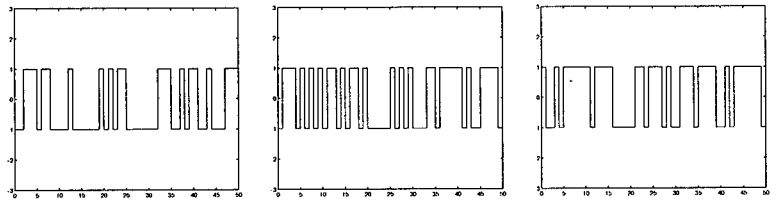


(a) (b) (c)

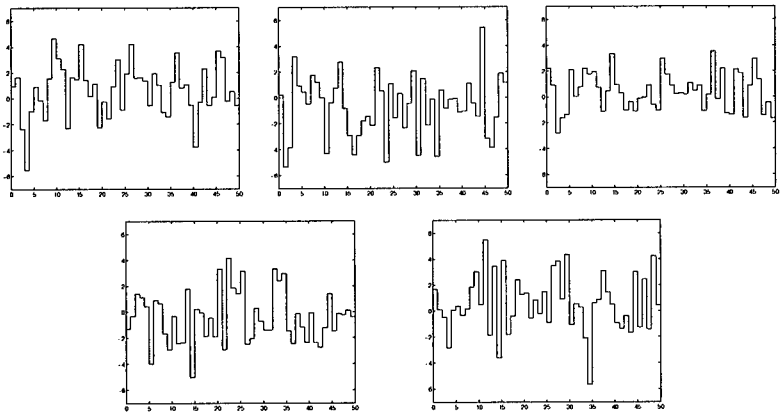
Figure 3: The eye pattern of recovered signals: (a) $z_1(k)$; (b) $z_2(k)$; (c) $z_3(k)$.

5 CONCLUSION

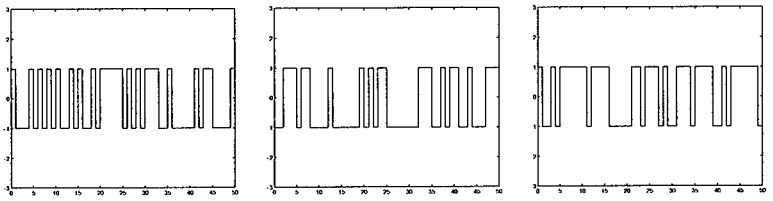
We have presented a new adaptive scheme to blind separation of source signals from their convolutive mixtures. Fundamental theoretical results and implementation have been presented. Under mild conditions, we have shown that i.i.d. source signals



(a)



(b)



(c)

Figure 4: (a) Three original source signals, $s_1(k)$, $s_2(k)$, $s_3(k)$; (b) Five received signals, $x_1(k)$, $x_2(k)$, $x_3(k)$, $x_4(k)$, $x_5(k)$; (c) Three recovered signals, $z_1(k) = -s_2(k)$, $z_2(k) = s_1(k)$, $z_3(k) = s_3(k)$

can be recovered by the generalized zero-forcing equalizer and instantaneous blind separation. A linear feedback network with associated anti-Hebbian learning has been constructed to perform spatio-temporal decorrelation. Computer simulation experiments demonstrated validity and high performance of our proposed approach.

References

[1] S. Amari, A. Cichocki, and H. H. Yang, "A new learning algorithm for blind signal separation," in **Advances in Neural Information Processing Systems**, Vol. 8, pp. 757-763, MIT press, 1996.

[2] S. Amari, S. C. Douglas, A. Cichocki, and H. H. Yang, "Multichannel blind deconvolution and equalization using the natural gradient," in **SPAWC** (Paris France), pp. 101-104, April 1997.

[3] A. Bell and T. Sejnowski, "An information maximisation approach to blind separation and blind deconvolution," **Neural Computation** 7, pp. 1129-1159, 1995.

[4] J. -F. Cardoso and B. H. Laheld, "Equivariant adaptive source separation," **IEEE Trans. on Signal Processing,** Vol. 44, pp. 3017-3030, 1996.

[5] D. C. B. Chan, S. J. Godsill, and P. J. W. Rayner, "Blind signal separation by output decorrelation," in **NIPS Workshop on Blind Signal Processing and its Applications** (Aspen Colorado), December 1996.

[6] S. Choi and R. Liu, "A direct adaptive blind equalizer for multi-channel transmission,' in **NIPS Workshop on Blind Signal Processing and its Applications** (Aspen Colorado), December 1996.

[7] S. Choi, H. Luo, and R. Liu, "An adaptive system for direct multi-channel blind equalization," in **SPAWC** (Paris France), pp. 85-88, April 1997.

[8] S. Choi and A. Cichocki, "A linear feedforward network with lateral feedback connections for blind source separation," in **IEEE Workshop on Higher-order Statistics,** (Banff, Canada), July 1997. (in print)

[9] A. Cichocki, R. Unbehauen, L. Moszczynski, and E. Rummert, "A new on-line adaptive learning algorithm for blind separation of source signals," in **Int. Symp. on Artificial Neural Networks**, pp. 406-411, 1994.

[10] A. Cichocki, S. Amari, and J. Cao, "Blind separation of delayed and convolved signals with self-adaptive learning rate," in **Proc. Int. Symp. on Nonlinear Theory and Applications (NOLTA)** (Kochi, Japan), pp. 229-232, 1996.

[11] D. Van Compernolle and S. Van Gerven, "On the use of decorrelation in scalar signal separation," in **ICASSP,** pp. 57-60, 1994.

[12] N. Delfosse and P. Loubaton, "Adaptive blind separation of convolutive mixtures," in **ICASSP,** pp. 2940-2943, 1996.

[13] S. Icart and R. Gautier, "Blind separation of convolutive mixtures using second and fourth order moments," in **ICASSP,** pp. 3018-3021, 1996.

[14] C. Jutten and J. Herault, "Blind separation of sources, part I: an adaptive algorithm based on neuromimetic architecture," **Signal Processing**, Vol. 24, pp. 1-10, 1991.

[15] H. N. Thi and C. Jutten, "Blind source separation for convolutive mixtures," **Signal Processing,** Vol. 45, pp. 209-229, 1995.

[16] M. Girolami and C. Fyfe, "A temporal model of linear anti-Hebbian learning," **Neural Processing Letters,** pp. 139-148, December 1996.

[17] R. H. Lambert and C. L. Nikias, "Bussgang methods for separation of multipath mixtures," in **NIPS Workshop on Blind Signal Processing and its Applications** (Aspen Colorado), December 1996.

[18] T. Lee, A. Bell, and R. Lambert, "Blind separation of delayed and convolved sources," in **Advances in Neural Information Processing Systems,** MIT press, 1997. (in print)

[19] T. Lee, A. Bell, and R. Orglmeister, "Blind source separation of real world signals," in ICNN, 1997.

[20] R. Liu and G. Dong, "A fundamental theorem for multi-channel blind equalization," **IEEE Trans. on Circuits and Systems I: Fundamental Theory and Applications.** (to appear)

[21] R. Liu, "A fundamental theorem for direct blind equalization," in **ISCAS**, 1997. (to appear)

[22] J. L. Massey and M. K. Sain, "Inverses of linear sequential circuits," **IEEE Trans. Computers,** Vol. C-17, pp. 330-337, 1968.

[23] J. Principe, C. Wang, and H. Wu, "Temporal decorrelation using teacher forcing anti-Hebbian learning and its application in adaptive blind source separation," in **NIPS Workshop on Blind Signal Processing and its Applications** (Aspen Colorado), December 1996.

[24] K. Torkkola, "Blind separation of convolved sources based on information maximisation," in **Proc. IEEE Workshop on Neural Networks and Signal Processing** (Kyoto, Japan), pp. 423-432, September 1996.

[25] D. Yellin and E. Weinstein, "Criteria for multichannel signal separation," **IEEE Trans. SP,** Vol. 42, pp. 2158-2168, 1994.

MULTICHANNEL BLIND SEPARATION AND DECONVOLUTION OF SOURCES WITH ARBITRARY DISTRIBUTIONS

Scott C. Douglas[1], Andrzej Cichocki[2], and Shun-ichi Amari[2]

[1]Department of Electrical Engineering, University of Utah
Salt Lake City, Utah 84112 USA

[2]Brain Information Processing Group, Frontier Research Program, RIKEN
Wako-shi, Saitama 351-01 JAPAN

Abstract– **Blind deconvolution and separation of linearly mixed and convolved sources is an important and challenging task for numerous applications. While several recently-developed algorithms have shown promise in these tasks, these techniques may fail to separate signal mixtures containing both sub- and super-Gaussian-distributed sources. In this paper, we present a simple and efficient extension of a family of algorithms that enables the separation and deconvolution of mixtures of arbitrary non-Gaussian sources. Our technique monitors the statistics of each of the outputs of the separator using a rigorously-derived sufficient criterion for stability and then selects the appropriate nonlinearity for each channel such that local convergence conditions of the algorithm are satisfied. Extensive simulations show the validity and efficiency of our method to blindly extract mixtures of arbitrary-distributed source signals.**

I. INTRODUCTION

Blind signal separation is useful for numerous problems in biomedical signal analysis, acoustics, communications, and signal and image processing. In blind source separation of instantaneous signal mixtures, a set of measured signals $\{x_i(k)\}$, $1 \leq i \leq n$ is assumed to be generated from a set of unknown stochastic independent sources $\{s_i(k)\}$, $1 \leq i \leq m$, $m \leq n$ as

$$\mathbf{x}(k) = \mathbf{H}\mathbf{s}(k), \tag{1}$$

where $\mathbf{x}(k) = [x_1(k) \cdots x_n(k)]^T$, $\mathbf{s}(k) = [s_1(k) \cdots s_m(k)]^T$, and $\mathbf{H}$ is an $(n \times m)$-dimensional matrix of unknown mixing coefficients $\{h_{ij}\}$. The measured sensor signals are processed by a linear single-layer feed-forward network as

$$\mathbf{y}(k) = \mathbf{W}(k)\mathbf{x}(k), \tag{2}$$

0-7803-4256-9/97/$10.00

where $\mathbf{y}(k) = [y_1(k) \cdots y_m(k)]^T$ and $\mathbf{W}(k)$ is an $(m \times n)$-dimensional synaptic weight matrix. Ideally, $\mathbf{W}(k)$ is adjusted iteratively such that

$$\lim_{k \to \infty} \mathbf{W}(k)\mathbf{H} = \mathbf{PD}, \tag{3}$$

where $\mathbf{P}$ is an $(m \times m)$-dimensional permutation matrix with a single unity entry in any of its rows or columns and $\mathbf{D}$ is a diagonal nonsingular matrix.

Recently, several simple, efficient, and robust iterative algorithms for adjusting $\mathbf{W}(k)$ have been proposed for the blind signal separation task [1]–[12]. Such methods use higher-order statistical information about the source signals to iteratively adjust the coefficient matrix $\mathbf{W}(k)$. In this paper, we consider one class of on-line adaptive algorithms given by [1]

$$\mathbf{W}(k+1) = \mathbf{W}(k) + \eta(k) \left[\mathbf{I} - \mathbf{f}(\mathbf{y}(k))\mathbf{y}^T(k)\right] \mathbf{W}(k), \tag{4}$$

where $\mathbf{f}(\mathbf{y}(k)) = [f_1(y_1(k)) \cdots f_m(y_m(k))]^T$. The optimal forms of the nonlinear functions $\{f_i(y)\}$ can be shown to be dependent on the statistics of the source signals [2, 7, 8]. For example, if the signal mixture consists of sub-Gaussian sources with negative kurtoses, the choices $f_i(y) = f_N(y) = |y|^p \mathrm{sgn}(y)$ for $p = \{2, 3, \ldots\}$ provide adequate separation capabilities. For mixtures of super-Gaussian sources with positive kurtoses, the choice $f_i(y) = f_P(y) = \tanh(\alpha y)$ with $\alpha > 0$ can be used [3, 11, 12].

A related task to blind signal separation is that of multichannel signal deconvolution, in which $\mathbf{x}(k)$ is assumed to be produced from $\mathbf{s}(k)$ as

$$\mathbf{x}(k) = \sum_{p=-\infty}^{\infty} \mathbf{H}_p \mathbf{s}(k-p), \tag{5}$$

where $\mathbf{H}_p$ is an $(n \times m)$-dimensional matrix of mixing coefficients at lag p. The goal is to calculate a vector $\mathbf{y}(k)$ of possibly scaled and/or delayed estimates of the source signals in $\mathbf{s}(k)$ from $\mathbf{x}(k)$ using a causal linear filter given by

$$\mathbf{y}(k) = \sum_{p=0}^{L} \mathbf{W}_p(k)\mathbf{x}(k-p), \tag{6}$$

where the $(m \times n)$-dimensional matrices $\{\mathbf{W}_p(k)\}$, $0 \leq p \leq L$ contain the coefficients of the multichannel filter. One algorithm that can be used in this task is described in [5, 6] and is given by

$$\mathbf{W}_p(k+1) = \mathbf{W}_p(k) + \eta(k) \left[\mathbf{W}_p(k) - \mathbf{f}(\mathbf{y}(k-L))\mathbf{u}^T(k-p)\right], \tag{7}$$

where the n-dimensional vector $\mathbf{u}(k)$ is computed as

$$\mathbf{u}(k) = \sum_{q=0}^{L} \mathbf{W}_{L-q}^T(k)\mathbf{y}(k-q). \tag{8}$$

This algorithm reduces to that in (4) for $L = 0$.

Although simulations have indicated that these algorithms are successful at separating and deconvolving linearly-mixed signals, they require knowledge about the statistics of the source signals to function properly. In particular, it must be known *a priori* whether the source signals are sub-Gaussian or super-Gaussian so that the nonlinearities $f_i(y)$ can be properly chosen. Even worse, if the measured signals $x_i(k)$ contain mixtures of both sub-Gaussian (e.g. digital data) and super-Gaussian (e.g. speech) sources, then these algorithms may fail to separate these signals reliably.

In this paper, we propose modifications to the algorithms in (4) and (7) that enable sources from arbitrary non-Gaussian distributions to be extracted from measurements of the mixed signals. Our methods use simple sufficient conditions for algorithm stability that are based on the necessary stability conditions originally derived by Amari *et al* [3]. Our computationally-simple algorithms employ time-varying nonlinearities in the coefficient updates that are selected from a family of fixed nonlinearities at each iteration to best satisfy our sufficient stability conditions. Simulations show the excellent and robust convergence behavior of the proposed methods in separating mixtures of sub- and super-Gaussian sources.

II. CRITERIA FOR ALGORITHM STABILITY

The modified algorithms for separation of sources with arbitrary distributions are based on the stability analysis of (4) that is described in [3]. For brevity and simplicity of discussion, we only consider (4) and outline the necessary extensions of the analysis that are needed to develop our modified algorithms, although we later apply the results to the multichannel deconvolution method in (7).

The algorithm in (4) can be derived as an iterative stochastic minimization procedure for the cost function

$$\phi(\mathbf{W}(k)) \;=\; -\frac{1}{2}\log(\det(\mathbf{W}(k)\mathbf{W}(k)^T)) - \sum_{i=1}^{m} E\{\log p_i(y_i(k))\}, \quad (9)$$

where $E\{\cdot\}$ denotes statistical expectation and $-d\log p_i(y)/dy = f_i(y)$. If $p_i(y)$ is the actual probability distribution of the source extracted at the ith output, then $\phi(\mathbf{W}(k))$ represents the negative of the maximum likelihood cost function [2, 7]. The procedure in (4) represents the natural gradient method for minimizing (9) iteratively using signal measurements. For details on the general form of the natural gradient search method, the reader is referred to [1].

In [3], the stability of (4) is analyzed by studying the expected value of the Hessian of the cost function, denoted as $E\{d^2\phi(\mathbf{W}(k))/dw_{ij}(k)dw_{pq}(k)\}$,

in the vicinity of a source separation solution. Here, $w_{ij}(k)$ is the (i,j)th element of $\mathbf{W}(k)$, which in [3] is assumed to be a square matrix ($m = n$). In what follows, we remove this restriction. In analogy with the results of [3], it is simpler to consider the form of $d^2\phi(\mathbf{W}(k))$ in terms of the modified coefficient differential

$$d\mathbf{X}(k) = d\mathbf{W}(k)\mathbf{W}^T(k)(\mathbf{W}(k)\mathbf{W}^T(k))^{-1}, \tag{10}$$

such that

$$d\mathbf{X}(k)\mathbf{y}(k) = d\mathbf{W}(k)\mathbf{x}(k). \tag{11}$$

Note that the natural gradient method automatically performs its search in the coefficient space spanned by $d\mathbf{X}(k)$, so that the coefficient updates remain in the original column space of $\mathbf{W}(0)$ for all k. We can then represent the differential $d^2\phi(\mathbf{W}(k))$ in terms of the elements of $d\mathbf{X}(k)$ as

$$\begin{aligned} d^2\phi(\mathbf{W}(k)) &= \mathbf{y}^T(k)d\mathbf{X}^T(k)\mathbf{F}_d''(\mathbf{y}(k))d\mathbf{X}(k)\mathbf{y}(k) \\ &\quad + \mathbf{f}^T(\mathbf{y}(k))d\mathbf{X}(k)d\mathbf{X}(k)\mathbf{y}(k), \end{aligned} \tag{12}$$

where $\mathbf{F}_d''(\mathbf{y}(k))$ is a diagonal matrix whose (i,i)th entry is $f_i''(y_i(k))$.

As is shown in [3], the expectations of the terms on the RHS of (12) are

$$\begin{aligned} E\{\mathbf{y}^T(k)d\mathbf{X}^T(k)\mathbf{F}_d''(\mathbf{y}(k))d\mathbf{X}(k)\mathbf{y}(k)\} &= \sum_{i=1}^{m}\sum_{j=1,j\neq i}^{m}\sigma_i^2(k)\kappa_j(k)[dx_{ji}(k)]^2 \\ &\quad + \sum_{i=1}^{m}\mu_i(k)[dx_{ii}(k)]^2 \end{aligned} \tag{13}$$

$$E\{\mathbf{f}^T(\mathbf{y}(k))d\mathbf{X}(k)d\mathbf{X}(k)\mathbf{y}(k)\} = \sum_{i=1}^{m}\sum_{j=1}^{m}\rho_i(k)dx_{ij}(k)dx_{ji}(k), \tag{14}$$

where $\mu_i(k)$, $\sigma_i^2(k)$, $\kappa_j(k)$, and $\rho_i(k)$ are defined as

$$\mu_i(k) = E\{y_i^2(k)f_i''(y_i(k))\}, \qquad \sigma_i^2(k) = E\{y_i^2(k)\}, \tag{15}$$

$$\kappa_j(k) = E\{f_j''(y_j(k))\}, \quad \text{and} \quad \rho_i(k) = E\{y_i(k)f(y_i(k))\}, \tag{16}$$

respectively, and where it has been assumed that $y_i(k)$ and $y_j(k)$ are independent for $i \neq j$. Thus, the expected value of the Hessian is

$$\begin{aligned} E\{d^2\phi(\mathbf{W}(k))\} &= \sum_{i=1}^{m}\sum_{j=1,j\neq i}^{m}\sigma_i^2\kappa_j(k)[dx_{ji}(k)]^2 + \rho_i(k)dx_{ij}(k)dx_{ji}(k) \\ &\quad + \sum_{i=1}^{m}[\mu_i(k)+\rho_i(k)][dx_{ii}(k)]^2. \end{aligned} \tag{17}$$

For stability, $E\{d^2\phi(\mathbf{W}(k))\}$ must be positive for all possible values of $dx_{ij}(k)$. By examining the RHS of (17), one can obtain the following necessary and sufficient stability conditions on $f_i(y)$ for all $1 \leq i < j \leq m$:

$$\kappa_i(k) > 0 \tag{18}$$

$$\mu_i(k) + \rho_i(k) > 0 \tag{19}$$

$$\sigma_i^2(k)\kappa_i(k)\sigma_j^2(k)\kappa_j(k) > \frac{1}{4}[\rho_i(k) + \rho_j(k)]^2. \tag{20}$$

The conditions in (18) and (19) are satisfied in practice for any odd nondecreasing function $f_i(y) = -f_i(-y)$. However, the condition in (20) is difficult to calculate in practice, as it involves $m(m-1)/2$ different combinations for $1 \leq i < j \leq m$. For this reason, we consider a sufficient stability criterion of the form

$$\sigma_i^2(k)\kappa_i(k)\sigma_j^2(k)\kappa_j(k) > \gamma^2(k)\rho_i(k)\rho_j(k), \tag{21}$$

where $\gamma(k)$, $\gamma(k) > 0$ satisfies for all $1 \leq i < j \leq m$ the inequality

$$\gamma^2(k)\rho_i(k)\rho_j(k) \geq \frac{1}{4}\left[\rho_i(k) + \rho_j(k)\right]^2. \tag{22}$$

After some algebra, we find that the smallest value of $\gamma(k)$ satisfying (22) is

$$\gamma(k) = \frac{1}{2}\sqrt{2 + \frac{\rho_{max}(k)}{\rho_{\min}(k)} + \frac{\rho_{min}(k)}{\rho_{max}(k)}}, \tag{23}$$

where $\rho_{max}(k)$ and $\rho_{min}(k)$ are the maximum and minimum values of $\rho_i(k)$ for $1 \leq i \leq m$, respectively. For this value of $\gamma(k)$, we can guarantee the stability of the algorithm in (4) if, for all $1 \leq i \leq m$,

$$\sigma_i^2(k)\kappa_i(k) - \gamma(k)\rho_i(k) > 0. \tag{24}$$

Note that all values of $\rho_i(k)$ converge to one as the coefficients converge to a separating solution due to the normalizing condition $E\{\mathbf{f}(\mathbf{y}(k))\mathbf{y}^T(k)\} = \mathbf{I}$, such that $\gamma(k) \approx 1$ near convergence.

III. THE ALGORITHM MODIFICATION

We now describe the modified algorithms that are based on the stability criteria of the last section. It is known that the algorithm in (4) can determine a separating solution for $\mathbf{W}(k)$ if a set of nonlinearities $\{f_i(y)\}$, $1 \leq i \leq m$ can be properly chosen. For this reason, our modified algorithms employ a time-varying vector nonlinearity $\mathbf{f}_k(\mathbf{y}(k)) = [f_{1k}(y_1(k)) \cdots f_{mk}(y_m(k)]^T$,

where $f_{ik}(y)$ is chosen to be one of two nonlinearities $f_N(y)$ and $f_P(y)$ that are optimized for sub- and super-Gaussian source separation tasks, respectively. To select $f_{ik}(y)$ at time k, we form the time-averaged estimates

$$\sigma_i(k) = (1-\delta)\sigma_i(k-1) + \delta|y_i(k)|^2 \quad (25)$$

$$\kappa_{ir}(k) = (1-\delta)\kappa_{ir}(k-1) + \delta f_r'(y_i(k)) \quad (26)$$

$$\rho_{ir}(k) = (1-\delta)\rho_{ir}(k-1) + \delta y_i(k) f_r(y_i(k)) \quad (27)$$

for $r = \{N, P\}$ and $1 \leq i \leq m$, where δ is a small positive parameter. Then, $f_{ik}(y)$ at time k is selected as

$$f_{ik}(y) = \begin{cases} f_N(y) & \text{if } \sigma_i^2(k)\kappa_{iN}(k) - \gamma(k)\rho_{iN}(k) > \sigma_i^2(k)\kappa_{iP}(k) - \gamma(k)\rho_{iP}(k) \\ f_P(y) & \text{if } \sigma_i^2(k)\kappa_{iN}(k) - \gamma(k)\rho_{iN}(k) < \sigma_i^2(k)\kappa_{iP}(k) - \gamma(k)\rho_{iP}(k) \end{cases} \quad (28)$$

where $\gamma(k)$ is computed at infrequent intervals from past estimates of $\rho_i(k)$. With these choices, the resulting vector $\mathbf{f}_k(\mathbf{y}(k))$ is used in place of $\mathbf{f}(\mathbf{y}(k))$ to adjust the coefficient matrix in (4). In simulations, it was found that the value of $\gamma(k)$ did not vary significantly over time, and in fact, setting $\gamma(k)$ equal to one in (28) for all k appears to provide convergence of the algorithms to a separating solution.

It can be seen that as the coefficients of the system converge, the quantity $\sigma_i^2(k)\kappa_{ir}(k) - \rho_{ir}(k)$ becomes a reliable estimate of the left-hand-side of the inequality in (24) for $f_{ik}(y) = f_r(y)$. Extensive simulations have shown that, so long as a set of nonlinearity assignments exists such that the stability conditions in (18)–(20) are satisfied for one ordering of the extracted sources at the outputs, then (28) properly selects each $f_{ik}(y)$ over time to enable the system to reliably extract all source signals regardless of their distribution.

IV. SIMULATIONS

We now show the capabilities of our modified source separation algorithms via simulation. In our first example, we employ the signal separation method in (4) to separate ten instantaneously-mixed signals. In this case, the three signal sets $\{s_1(k), s_2(k), s_3(k), s_4(k)\}$, $\{s_5(k), s_6(k), s_7(k)\}$, and $\{s_8(k), s_9(k), s_{10}(k)\}$ are i.i.d. with Laplacian, uniform-$[-1, 1]$, and binary-$\{\pm 1\}$ distributions, respectively, where the Laplacian p.d.f. is given by $p_s(s) = 0.5e^{-|s|}$. Since the first and latter two distributions are super- and sub-Gaussian, respectively, the algorithms in [4, 8] cannot linearly separate these sources from an arbitrary mixture of them. We generate $\mathbf{x}(k)$ as in (1), where the entries of $\mathbf{H}$ are drawn from a uniform-$[0, 1]$ distribution. The values of h_{ij} to four decimal places are shown in Table 1. As is clear from the table, $\mathbf{H}$ exhibits no particular structure, and thus the extraction of the ten sources from the measured signals $\mathbf{x}(k)$ is a challenging task.

Table 1: The entries of $\mathbf{H}$ for the ten source separation example.

(i,j)	1	2	3	4	5	6	7	8	9	10
1	0.8424	0.2541	0.3297	0.6648	0.9273	0.4364	0.6019	0.8918	0.9564	0.8823
2	0.0710	0.0570	0.6562	0.4651	0.5970	0.9322	0.7337	0.7483	0.3582	0.6291
3	0.7041	0.4257	0.5772	0.7443	0.8824	0.9441	0.2775	0.4733	0.1009	0.3935
4	0.4146	0.3068	0.6455	0.9982	0.0223	0.9772	0.1033	0.1339	0.3141	0.9203
5	0.5751	0.6326	0.5300	0.4357	0.4652	0.5468	0.0358	0.1748	0.1137	0.0413
6	0.1854	0.4599	0.4425	0.1115	0.2274	0.9701	0.1367	0.6777	0.7476	0.6656
7	0.1480	0.2081	0.2611	0.2721	0.1238	0.7741	0.0425	0.4375	0.8457	0.7489
8	0.8016	0.2801	0.6381	0.0364	0.0919	0.6716	0.7553	0.7595	0.3628	0.0460
9	0.9349	0.1214	0.1404	0.0398	0.4975	0.9554	0.4872	0.2505	0.6368	0.7157
10	0.6022	0.2268	0.2975	0.8565	0.8663	0.9502	0.9896	0.4781	0.3873	0.2050

We apply the algorithm in (4), where $\mathbf{f}(\mathbf{y}(k)) = \mathbf{f}_k(\mathbf{y}(k))$ is adapted according to our method described in (25)–(28) with $\eta(k) = 0.005$, $\delta = 0.01$, $\mathbf{W}(0) = \mathbf{I}$, and

$$f_N(y) = |y|^3 \mathrm{sgn}(y) \quad \text{and} \quad f_P(y) = \tanh(10y), \tag{29}$$

corresponding to nonlinearities for separating sub- and super-Gaussian-distributed sources, respectively. Within each on-line nonlinearity selection procedure, we set $\gamma(k)$ to one for all k. From the outputs of the separation system, we compute the error vector $\mathbf{e}(k) = [e_1(k) \ \cdots \ e_{10}(k)]^T$ as

$$\mathbf{e}(k) \ = \ \mathbf{s}(k) - \mathbf{D}^{-1}\mathbf{P}^T\mathbf{y}(k), \tag{30}$$

where approximate versions of the permutation matrix $\mathbf{P}$ and scaling matrix $\mathbf{D}$ as introduced in (3) are obtained from $\mathbf{W}(k)$ and $\mathbf{H}$ at iteration $k = 10000$. Figure 1 shows these ten error signals. Since each error signal decreases to a small value after a sufficient number of iterations, all of the sources are reliably extracted using our modified algorithm. Figure 2 plots the performance factor $\psi(k)$ defined as

$$\psi(k) \ = \ \frac{||\mathbf{P}^T\mathbf{W}(k)\mathbf{H}||_E^2}{||[\mathbf{P}^T\mathbf{W}(k)\mathbf{H}]_d||_E^2} - 1, \tag{31}$$

where $||\cdot||_E$ denotes the matrix Euclidean norm and where $[\mathbf{Q}]_d$ is a diagonal matrix whose $(i,i)th$ entry is q_{ii}. As can be seen, the value of $\psi(k)$ decreases to approximately 0.0168 in steady-state, indicating that the system has adequately separated the ten sources. Moreover, a careful examination of the nonlinearities chosen for each extracted output indicate that the appropriate stabilizing nonlinearity $f_N(y)$ or $f_P(y)$ was eventually selected for each output signal.

We now combine the algorithm modification in (25)–(28) with the blind deconvolution and source separation technique in (7) and apply the resulting system to a three-source separation problem. In this case, the three sources are chosen to be i.i.d. Laplacian-, uniform-, and binary-distributed, respectively, and the convolutive mixture model is given by

$$\mathbf{x}(k) \ = \ \mathbf{A}_1\mathbf{x}(k-1) + \mathbf{A}_2\mathbf{x}(k-2) + \mathbf{B}_0\mathbf{s}(k) + \mathbf{B}_1\mathbf{s}(k-1), \tag{32}$$

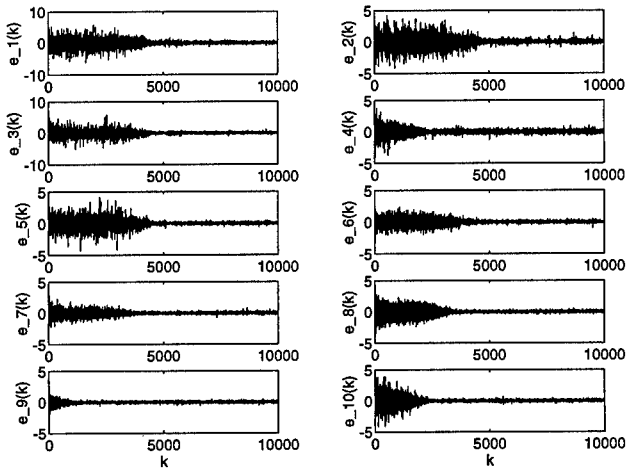


Figure 1: The ten error signals for the first source separation example.

where the (3×3) matrices $\mathbf{A}_1$, $\mathbf{A}_2$, $\mathbf{B}_0$, and $\mathbf{B}_1$ are given by

$$\mathbf{A}_1 = \begin{bmatrix} -0.1 & 0.3 & -0.1 \\ -0.2 & 0.2 & -0.3 \\ -0.5 & -0.2 & 0.4 \end{bmatrix}, \qquad \mathbf{A}_2 = \begin{bmatrix} 0.04 & 0.02 & 0.03 \\ 0.08 & 0.04 & 0 \\ 0.03 & 0.06 & 0.06 \end{bmatrix}, \quad (33)$$

$$\mathbf{B}_0 = \begin{bmatrix} 0.02 & 0.07 & 0.05 \\ 0 & 0.09 & 0.08 \\ 0.07 & 0.04 & 0 \end{bmatrix}, \quad \text{and} \quad \mathbf{B}_1 = \begin{bmatrix} 0.1 & 0 & 0.4 \\ 0.5 & 0.4 & 0.7 \\ 0.7 & 0.1 & 0.6 \end{bmatrix}. \quad (34)$$

Figures 3(a)-(c) and (d)-(f) show the vector sequences $\mathbf{s}(k)$ and $\mathbf{x}(k)$ in this case. The deconvolution system with time-varying nonlinearities was applied to these signals, in which $L = 6$, $\eta(k) = 0.0005$, $\delta = 0.001$, $\mathbf{W}_p(0) = \mathbf{I}\delta(p-3)$, and $\gamma(k)$ was set to one for all k within the nonlinearity selection procedures. Shown in 3(g)-(i) are the error signals $e_1(k) = s_1(k) - y_3(k)/d_{33}$, $e_2(k) = s_2(k) - y_2(k)/d_{22}$, and $e_3(k) = s_3(k) - y_1(k)/d_{11}$, where d_{11}, d_{22}, and d_{33} are appropriate scaling factors. As can be seen, the errors decrease to small values for each extracted output, and the signal-to-noise ratios for the three extracted outputs were empirically found to be 10.0, 7.4, and 15.2 dB, respectively.

Figure 4 shows the actual value of $\gamma(k) - 1$ on a logarithmic scale for the second source separation example. Starting from an initial value of $\gamma(1) = 2.94$, the value of $\gamma(k)$ gradually approaches unity over time. These results, combined with the successful separation capabilities of the modified systems, indicate that setting $\gamma(k)$ to one within the nonlinearity selection procedures does not limit the overall capabilities of the systems.

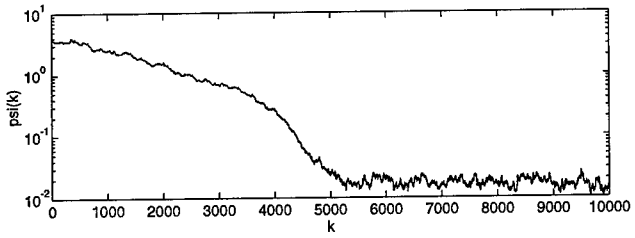


Figure 2: Performance factor $\psi(k)$ for the first source separation example.

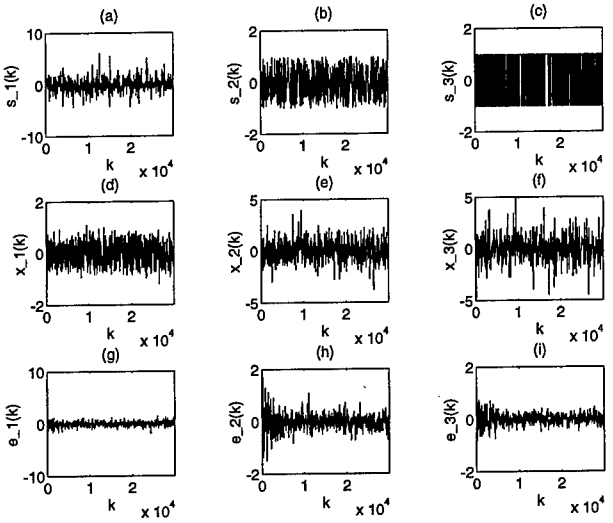


Figure 3: The three source signals ((a)-(c)), the three mixed signals ((d)-(f)), and the three error signals ((g)-(i)) for the convolutive-mixture source separation example.

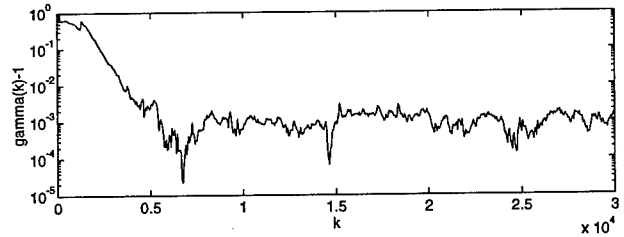


Figure 4: Evolution of $\gamma(k)-1$ for the convolutive-mixture source separation example.

V. CONCLUSIONS

In conclusion, we have described techniques for selecting the nonlinearities within blind source separation and deconvolution algorithms to enable the separation of sources with arbitrary distributions. The proposed methods can be easily implemented in an on-line setting. Simulations applying the techniques to instantaneous mixture separation and to multichannel deconvolution and source separation indicate the ability of the methods to accurately separate signal mixtures containing both sub- and super-Gaussian sources.

REFERENCES

[1] S. Amari, "Natural gradient works efficiently in learning," submitted to *Neural Computation.*

[2] S. Amari and J.-F. Cardoso, "Blind source separation — semi-parametric statistical approach," to appear in *IEEE Trans. Signal Processing.*

[3] S. Amari, T.-P. Chen, and A. Cichocki, "Stability analysis of adaptive blind source separation," to appear in *Neural Networks.*

[4] S. Amari, A. Cichocki, and H.H. Yang, "A new learning algorithm for blind signal separation," *Adv. Neural Inform. Proc. Sys. 8* (Cambridge, MA: MIT Press, 1996), pp. 757-763.

[5] S. Amari, S.C. Douglas, A. Cichocki, and H.H. Yang, "Multichannel blind deconvolution using the natural gradient," *Proc. IEEE Workshop Signal Proc. Adv. Wireless Comm.* (Piscataway, NJ: IEEE, 1997), pp. 101-104.

[6] S. Amari, S.C. Douglas, A. Cichocki, and H.H. Yang, "Novel on-line adaptive learning algorithms for blind deconvolution using the natural gradient approach," presented at *11th IFAC Symp. Syst. Ident.*, Kitakyushu, Japan, July 1997.

[7] A.J. Bell and T.J. Sejnowski, "An information maximization approach to blind separation and blind deconvolution," *Neural Computation,* vol. 7, no. 6, pp. 1129-1159, Nov. 1995.

[8] J.-F. Cardoso and B. Laheld, "Equivariant adaptive source separation," *IEEE Trans. Signal Processing,* vol. 44, pp. 3017-3030, Dec. 1996.

[9] A. Cichocki, R. Unbehauen, and E. Rummert, "Robust learning algorithm for blind separation of signals," *Electron. Lett.,* vol. 30, no. 17, pp. 1386-1387, Aug. 1994.

[10] A. Cichocki, S. Amari, M. Adachi, and W. Kasprzak, "Self-adaptive neural networks for blind separation of sources," *Proc. IEEE Int. Symp. Circ. Syst.,* vol. 2 (New York: IEEE, 1996), pp. 157-160.

[11] P. Comon, C. Jutten, and J. Herault, "Blind separation of sources, II: Problem statement," *Signal Processing,* vol. 24, no. 1, pp. 11-20, July 1991.

[12] C. Jutten and J. Herault, "Blind separation of sources, I: An adaptive algorithm based on neuromimetic architecture," *Signal Processing,* vol. 24, no. 1, pp. 1-10, July 1991.

RECURRENT CANONICAL PIECEWISE LINEAR NETWORK: THEORY AND APPLICATION

Xiao Liu and Tülay Adalı
Department of Computer Science and Electrical Engineering
University of Maryland Baltimore County, Baltimore, MD 21250
{xliu, adali}@engr.umbc.edu

Abstract- Recurrent Canonical Piecewise Linear (RCPL) network is defined by combining the canonical piecewise linear function with the autoregressive moving average (ARMA) model such that an augmented input space is partitioned into regions where an ARMA model is used in each. Properties of RCPL network are discussed. Particularly, it is shown that RCPL function is a contractive mapping and is stable in the sense of bounded input and bounded output stability. By generalizing Donoho's minimum entropy deconvolution approach [5] to the nonlinear case, it is shown that RCPL network can achieve blind equalization. RCPL network is applied to both supervised and blind equalization and results are presented to show that it is computationally efficient and with a very simple structure, can deliver highly satisfactory performance.

I. INTRODUCTION

Nonlinear techniques offer great potential to improve on the existing methods based on the assumption of linearity, and in certain cases, to deal with problems for which linear techniques have proven ineffective. Among the approaches proposed for nonlinear signal processing, polynomial or Volterra filters [12] can yield a small asymptotical probability of error if sufficiently high order polynomials are used, but they will also, in general, converge very slowly. Neural network structures, such as the multilayer perceptron [6] and the radial basis functions [3] have been proposed as an alternative to the nonlinear approximation problem, but in communications applications such as equalization that requires real-time processing capability, their relatively slow convergence characteristics have been the main consideration. The attractiveness of piecewise linear models on the other hand, stems from the fact that while they allow use of a variety of analysis and development tools that are linear, they are also good approximators of functions that are highly nonlinear. They have been effectively used in control engineering [13], nonlinear circuit analysis [9], and in channel equalization [2]. A special class of piecewise linear structures, canonical piecewise linear (CPL) models, employ a global linear model in a partitioned domain space rather than using individual linear models in each partition as does the piecewise linear model. Hence they greatly reduce the parameter storage requirement of the piecewise linear model and gain scheduling model [14], one of the major problems in the

0-7803-4256-9/97/$10.00

implementation of piecewise linear filters. We can use standard linear adaptive filtering techniques to perform training tasks on a piecewise linear filter and can easily incorporate known statistical information into the network structure.

We have introduced recurrent CPL function (RCPL) by combining the CPL function with the autoregressive moving average (ARMA) model [10]. Hence, RCPL mapping is the process of finding partition boundaries of a sample space in an augmented domain space where an ARMA model is used for approximation in each partitioned region. The proposed RCPL network offers the advantages of both the CPL and ARMA models. Specifically, RCPL network offers the following benefits: (1) it makes use of standard linear adaptive filtering techniques to perform training tasks and allows for efficient selection of the partition boundaries; (2) it offers savings in computation time and implementation cost, especially when modeling highly nonlinear functions; (3) because of its piecewise linear nature, it is easy to incorporate known statistical information into the network structure; (4) since RCPL network also employs feedback, it has a distinct dynamic behavior which is much more powerful than that attained by the use of finite duration impulse response feedforward structures. In this paper, we will first review the definition and approximation ability of RCPL network. Then, we discuss the properties of RCPL network. Finally, we consider application of recurrent canonical piecewise linear (RCPL) network to both supervised and blind equalization.

II. RECURRENT PIECEWISE LINEAR PIECEWISE NETWORK

Before giving the formal definition of a recurrent canonical piecewise linear representation, first, we present the definition of a canonical piecewise linear (CPL) function. CPL network is initially introduced for nonlinear circuit analysis [4]. CPL structures provide a desirable compromise between the approximation ability of nonlinear models and the efficiency and theoretical accessibility of the linear domain, and reduce the parameter storage requirement of piecewise linear models considerably by employing a global linear representation.

The CPL function is defined as [4]:

Definition 1 (*Canonical Piecewise Linear Function*): A piecewise linear function f: $D \rightarrow Q$, with a compact subset $D \subset R^N$ and compact subset $Q \subset R^M$, is called a canonical piecewise linear function, if it can be expressed by a *global* representation:

$$f(\mathbf{x}) = \mathbf{a} + \mathbf{B}\mathbf{x} + \sum_{i=1}^{\tau} \mathbf{c}_i \, |\langle \boldsymbol{\alpha}_i, \mathbf{x} \rangle + \beta_i| \tag{1}$$

where $\mathbf{B} \in R^{M \times N}, \mathbf{a}, \mathbf{c}_i \in R^M, \boldsymbol{\alpha}_i, \mathbf{x} \in R^N$ and $\beta_i \in R$.

On the other hand, a very general class of linear models used for prediction of a random process $x(n)$ is the class of ARMA models and a natural generalization of the linear ARMA model is the *nonlinear autoregressive moving average* (NARMA) model

$$x(n) = h(x(n-1), x(n-2), \cdots, x(n-\bar{p}_1), e(t-1), e(t-2), \cdots, e(n-\bar{p}_2))$$

where h is a nonlinear function and $e(n)$ is the input variable. Next, we introduce a RCPL function as an extension of CPL function by incorporating the NARMA model.

The RCPL function is defined as [10]:

Definition 2 (*Recurrent Canonical Piecewise-Linear Function*): A function f: $D_1 \times D_2 \times I \rightarrow Q$ with sample space $D_1 \subset R^N$, $D_2 \subset R^r$, index set I, and compact subset $Q \subset R^M$ is said to be a RCPL function if it can be expressed by the global representation:

$$f(\mathbf{x}(n), \mathbf{u}(n)) = \mathbf{a} + \mathbf{B}_0\mathbf{x}(n) + \mathbf{B}_1 f(\mathbf{x}(n-1), \mathbf{u}(n-1)) + \mathbf{B}_2\mathbf{u}(n) \qquad (2)$$

$$x_k(n) = a_k + \mathbf{b}_{1k}^T\mathbf{x}(n-1) + \mathbf{b}_{2k}^T f(\mathbf{x}(n-1), \mathbf{u}(n-1)) + \mathbf{b}_{3k}^T\mathbf{u}(n) + \sum_{i=1}^{\tau} c_{ki} \left| \langle \boldsymbol{\alpha}_{1ki}, \mathbf{x}(n-1) \rangle + \langle \boldsymbol{\alpha}_{2ki}, f(\mathbf{x}(n-1), \mathbf{u}(n-1)) \rangle + \langle \boldsymbol{\alpha}_{3ki}, \mathbf{u}(n) \rangle + \beta_{ki} \right| \qquad (3)$$

where $\mathbf{x}, \mathbf{b}_{1k}, \boldsymbol{\alpha}_{1ki} \in R^N$, $\mathbf{u}, \mathbf{b}_{3k}, \boldsymbol{\alpha}_{3ki} \in R^r$, $\mathbf{a}, \mathbf{b}_{2k}, \boldsymbol{\alpha}_{2ki}, \mathbf{B}_0 \in R^M$, $\mathbf{B}_1 \in R^{M \times N}$, $\mathbf{B}_2 \in R^{M \times r}$, $a_k, c_{ki}, \beta_{ki} \in R$, $k = 1, 2, \cdots, N$ and x_k is the kth element of vector $\mathbf{x}$. We refer to the structure defined by (2) and (3) as the *recurrent canonical piecewise linear network.*

From the definition, we see that the domain of RCPL function is partitioned into polyhedral regions where the function defines a linear ARMA model in each.

III. PROPERTIES OF RECURRENT PIECEWISE LINEAR NETWORK

Based on Definition 1, we show the approximation ability of CPL network by the following theorem:

Theorem 1: Let domain D be a compact space of dimension N and $\mathcal{F}$ be a set of canonical piecewise linear functions on D. Then, for any continuous function $\bar{f}$ on D, there exists a function $f \in \mathcal{F}$ such that $|f(\mathbf{x}) - \bar{f}(\mathbf{x})| < \epsilon$

for all $\mathbf{x} \in D$.

The proof of the theorem is given in [1]. By comparing definitions 1 and 2, we can see that RCPL filter is actually a special case of the CPL filter. The RCPL filter partitions the input signal space into finite disjoint regions and in each region, it can be represented by a FIR filter with infinite length. Therefore, the result presented in Theorem 1 also holds for the RCPL filter.

We can rewrite the RCPL network described by (2) and (3) as follows:

$$\begin{aligned} \bar{\mathbf{x}}(n) &= \bar{\mathbf{a}} + \bar{\mathbf{B}}_1 \bar{\mathbf{x}}(n-1) + \bar{\mathbf{B}}_2 \mathbf{u}(n) \\ &+ \sum_{i=1}^{\tau} c_i \left| \langle \bar{\boldsymbol{\alpha}}_{1i}, \mathbf{x}(n-1) \rangle + \langle \bar{\boldsymbol{\alpha}}_{2i}, \mathbf{u}(n) \rangle + \bar{\beta}_i \right| \end{aligned} \tag{4}$$

where $\bar{\mathbf{x}}(n) = (\mathbf{x}(n) \quad f(\mathbf{x}(n))^T$. Using this representation for RCPL, we prove the following:

Theorem 2: For the RCPL network defined by (2) and (3), assume that the input vector $\mathbf{u}(n)$ is bounded and the parameters satisfy the following condition: If there exists an $\varepsilon_0 \in (0,1)$ such that

$$||\bar{\mathbf{B}}_1|| + \sum_{i=1}^{\tau} ||c_i||\ ||\bar{\boldsymbol{\alpha}}_{1i}|| \leq 1 - \varepsilon_0 \tag{5}$$

then, there is a real number d, such that for all $K \geq d$, the ball $D(K) = \{\mathbf{x} : ||\mathbf{x}|| \leq K\}$ is invariant under (2) and (3).

Proof: Define $F(\bar{\mathbf{x}}(n-1)) \equiv \bar{\mathbf{x}}(n)$. From (4), we have

$$F(\bar{\mathbf{x}}(n-1)) = \bar{\mathbf{a}} + \bar{\mathbf{B}}_1 \bar{\mathbf{x}}(n-1) + \bar{\mathbf{B}}_2 \mathbf{u}(n) + \sum_{i=1}^{\tau} c_i \left| \langle \bar{\boldsymbol{\alpha}}_{1i}, \mathbf{x}(n-1) \rangle + \langle \bar{\boldsymbol{\alpha}}_{2i}, \mathbf{u}(n) \rangle + \bar{\beta}_i \right|$$

Since $\mathbf{u}(n)$ is bounded, there exists a real number r_0 such that for $||\bar{\mathbf{x}}(n-1)|| \geq r_0$, we have

$$\frac{||\bar{\mathbf{a}}|| + ||\bar{\mathbf{B}}_2||||\mathbf{u}(n)|| + \sum_{i=1}^{\tau} ||c_i||(||\bar{\boldsymbol{\alpha}}_{2i}||\ ||\mathbf{u}(n)|| + |\bar{\beta}_i|)}{||\bar{\mathbf{x}}(n-1)||} \leq \varepsilon_0 \tag{6}$$

therefore

$$\begin{aligned} &||\bar{\mathbf{a}} + \bar{\mathbf{B}}_1 \bar{\mathbf{x}}(n-1) + \bar{\mathbf{B}}_2 \mathbf{u}(n) + \sum_{i=1}^{\tau} c_i \left| \langle \bar{\boldsymbol{\alpha}}_{1i}, \bar{\mathbf{x}}(n-1) \rangle + \langle \bar{\boldsymbol{\alpha}}_{2i}, \mathbf{u}(n) \rangle + \bar{\beta}_i \right| || \\ &\leq ||\bar{\mathbf{a}}|| + (||\bar{\mathbf{B}}_1|| + \sum_{i=1}^{\tau} ||c_i||\ ||\bar{\boldsymbol{\alpha}}_{1i}||)||\bar{\mathbf{x}}(n-1)|| \\ &+ ||\bar{\mathbf{B}}_2||||\mathbf{u}(n)|| + \sum_{i=1}^{\tau} ||c_i||(||\bar{\boldsymbol{\alpha}}_{2i}||\ ||\mathbf{u}(n)|| + |\bar{\beta}_i|) \end{aligned}$$

Using equations (5) and (6), we have

$$\begin{aligned} ||F(\bar{\mathbf{x}}(n-1))|| &\leq (1-\varepsilon_0)||\bar{\mathbf{x}}(n-1)|| + \varepsilon_0||\bar{\mathbf{x}}(n-1)|| \\ &\leq ||\bar{\mathbf{x}}(n-1)|| \qquad \text{for} \quad ||\mathbf{x}(n-1)|| \geq r_0 \end{aligned} \tag{7}$$

On the other hand, because $F(\mathbf{x}(n-1))$ is continuous, there exists a constant K_0 such that

$$||F(\bar{\mathbf{x}}(n-1))|| \leq K_0 \qquad \text{for} \quad ||\bar{\mathbf{x}}(n-1)|| \leq r_0$$

Let $d = \max\{r_0, K_0\}$. Then, for every real number $K \geq d$, we have

$$||F(\bar{\mathbf{x}}(n-1))|| \leq K_0 \leq K \qquad \text{for every} \quad ||\bar{\mathbf{x}}(n-1)|| \leq r_0$$

and by using (7)

$$||F(\bar{\mathbf{x}}(n-1))|| \leq ||\bar{\mathbf{x}}(n-1)|| \leq K \qquad \text{for every} \quad r_0 \leq ||\bar{\mathbf{x}}(n-1)|| \leq K$$

Hence, $D(K)$ is invariant.

The above theorem states that the output of RCPL network described by bounded input if condition (5) is satisfied.

Theorem 3: The map which defines the RCPL network (2) and (3) is a contractive mapping if the condition given in (5) is satisfied.

Proof. Let

$$k(\mathbf{x}) = \bar{\mathbf{a}} + \bar{\mathbf{B}}_1\bar{\mathbf{x}} + \bar{\mathbf{B}}_2\mathbf{u}(n) + \sum_{i=1}^{\tau} c_i \left| \langle \bar{\boldsymbol{\alpha}}_{1i}, \mathbf{x} \rangle + \langle \bar{\boldsymbol{\alpha}}_{2i}, \mathbf{u}(n) \rangle + \bar{\beta}_i \right|$$

then,

$$\begin{aligned} k(\mathbf{x}_1) - k(\mathbf{x}_2) &= \bar{\mathbf{B}}_1(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2) \\ &+ \sum_{i=1}^{\tau} c_i \left(\left| \langle \bar{\boldsymbol{\alpha}}_{1i}, \mathbf{x}_1 \rangle + \langle \bar{\boldsymbol{\alpha}}_{2i}, \mathbf{u}(n) \rangle + \bar{\beta}_i \right| - \left| \langle \bar{\boldsymbol{\alpha}}_{1i}, \mathbf{x}_2 \rangle + \langle \bar{\boldsymbol{\alpha}}_{2i}, \mathbf{u}(n) \rangle + \bar{\beta}_i \right| \right) \end{aligned}$$

and

$$||k(\mathbf{x}_1) - k(\mathbf{x}_2)|| \leq \left(||\bar{\mathbf{B}}_1|| + \sum_{i=1}^{\tau} ||c_i||\, ||\bar{\boldsymbol{\alpha}}_{1i}|| \right) ||\mathbf{x}_1 - \mathbf{x}_2|| \leq (1-\varepsilon_0)||\mathbf{x}_1 - \mathbf{x}_2||$$

which shows that $k(\cdot)$ is a contractive mapping whenever (5) is satisfied. Thus, after receiving the input vector $\mathbf{u}(n)$, the network will always reach a unique equilibrium regardless of its initial state $\mathbf{x}_0$.

Theorems 1-3 state that under certain regularity conditions, the RCPL network is always stable in the sense of bounded input and bounded output stability, and can approximate any nonlinear function with arbitrary accuracy.

IV. BLIND EQUALIZATION BY RCPL NETWORK

Blind equalization refers to the problem of determining a transmitted symbol sequence in the presence of intersymbol interference (ISI) and noise without using a training sequence. Most of existing blind techniques such as the Bussgang algorithm, cyclic spectrum approach, polyspectra approach, etc., are based on the linear channel assumption. There are many cases, however, where this assumption is not true, as nonlinear devices significantly contribute to system degradation. One example is the digital satellite link, in which both the earth station and the satellite are equipped with amplifiers operated in a nonlinear region of the input-output characteristics for better exploitation of the power of the device. The use of the above blind techniques will suffer from a severe performance degradation for unknown nonlinear communication channels and hence it is very important to develop nonlinear blind equalization techniques. We show that nonlinear blind equalization can be achieved by matching the distribution of the channel input with that of the RCPL equalizer output.

To show the ability of RCPL network to achieve blind equalization, we first discuss some results based on the CPL network. We introduce the following: The nonlinear channel $h(\cdot)$ maps the input sequence $x(n) \in \Omega$ to $y(n) = h(x(n), x(n-1), \cdots, x(n-1-p))$ and the CPL equalizer aims to recover the input sequence by constructing a mapping $h_{eq} : D \rightarrow \Omega$ where $D \subset R^k$ and $\Omega \subset R$. Assume that the global system, cascade of the nonlinear channel $h(\cdot)$ and the CPL equalizer $h_{eq}(\cdot)$ is denoted by $\mathcal{T}$, and is modeled by a CPL network which divides the input space into m disjoint regions, $R_1, R_2, \cdots, R_m$, and in each region R_i, the CPL function given in (1) is equivalent to the following linear model:

$$M_i : \qquad \tilde{x}(n) = \sum_{j=1}^{k} w_{ij} x_j(n) \tag{8}$$

where $x_j(n) \equiv x(n-j+1)$ and $\tilde{x}(n)$ is the output of the equalizer.

We then make the following assumptions:

(i) Input sequence $\{x(n)\}$ is an i.i.d. random process.

(ii) The distribution of $x(n)$ is symmetric about zero with finite variance.

(iii) The mapping M_i, $i = 1, 2, \cdots, m$ is a one to one mapping, and $I_i \cap I_j = \emptyset$.

We then prove the following:

Theorem 4: Consider the global system $\mathcal{T}$ defined by (8) and that the assumptions (i)-(iii) are satisfied. If the distribution of $\{\tilde{x}(n)\}$ is the same as that of $x(n)$, then, the global system $\mathcal{T}$ is identity except for a possible delay and a sign factor.

The proof of Theorem 4 is given in [1]. As we explained in section III, the RCPL filter is a special case of the CPL filter. Therefore, the result presented in Theorem 4 is also true for the RCPL filter. Hence, any nonlinear channel can be represented as a RCPL function, and furthermore, if we use a RCPL network as an equalizer, then, the global system which consists of cascade of the the channel and the equalizer is still a RCPL function. Thus, for blind equalization, we can update the weights of RCPL equalizer such that the instantaneous distribution of the output $\tilde{x}(n)$ of the equalizer converges to the input distribution ν.

V. EXAMPLES

1. Blind Equalization

Several cost functions such as moment error cost function [8], Godard/Sato cost function [7], Vembu-Verdu's convex cost function [15] and partial likelihood cost function [1] can be used for distribution matching. Godard/Sato cost function is not a convex cost function, the derived blind algorithm may only find the local minimum. But, if proper initial weights are chosen, the algorithm can find the global minimum and the equalizer prediction error tends to zero. Thus, the resulting blind equalizer has larger stable margin. Vembu-Verdu's convex cost function can help the algorithm to find the global minimum for linear channel and the equalizer prediction error can quickly enter the decision boundary, however, it results in larger residual prediction error after convergence. In [11], a new equalizer structure is proposed by incorporating RCPL network with decision feedback. and a blind algorithm is presented based on both the Godard [7] and Vembu-Verdu's cost function [15] for the RCPL network based equalizer. We first use Vembu-Verdu's cost function in the learning process and then switch to the Godard cost function when the absolute gradient of Godard error changes are very slow. As an example, a nonlinear communication channel $g(n) = g_l(n)+0.1g_l(n)^2$ where the nonminimum phase multipath component is given by $g_l(n) = 0.9\delta(n)+\delta(n-1)$ is considered in [11]. The input $x(n)$ takes values form the binary set $\mathcal{S} = \{-1, 1\}$ and has a symmetric distribution. The simulation results given in [11] indicate that the blind algorithm which is derived based on combined Godard cost function and Vembu-Verdu's cost function exhibits good tradeoff in terms of robustness and achieving low equalizer prediction error. The developed blind algorithm is much faster than that derived based on the Godard cost function or the Vembu's cost function alone. Here, we only show the bit error curves of three cost functions in Figure 1. The RCPL network based decision feedback equalizer outperforms the linear decision feedback equalizer and CMA equalizer when equalizing a nonlinear channel.

2. Adaptive Equalization

A simple RCPL network and corresponding learning algorithm is given in [10].

Assume that $y(n)$ is the channel output corresponding to the transmitted signal $x(n)$ at time instant n. Let $\mathbf{y}(n) = [y(n), \cdots, y(n - M_1 + 1)]^T$ be the input vector of RCPL network and $\hat{x}(n)$ be the output of network trained to approximate $x(n)$. We choose the nonlinear function $f_k(\cdot)$ of the RCPL filter shown in Figure 2 as

$$f_k(z_k(n)) = |z_k(n) + 1| - |z_k(n) - 1|, \qquad k = 1, 2, \cdots, M_1$$

to equalize the following channel:

$$y(n) = y_l(n) + 0.02y_l(n)^2 + \eta(n) \tag{9}$$

where the multipath component is given by $y_l(n) = x(n) + 0.5x(n-1) + 0.4x(n-2)$, $x(n)$ is the input signal, $y(n)$ is the channel output, and $\eta(n)$ denotes the zero mean, white noise component.

Figures 3-5 compare the performance of the RCPL equalizer with that of the multilayer perceptron equalizer and the recurrent neural network (RNN) equalizer for 16-PAM, 8-PAM and 2-PAM signal transmission over the multipath channel of (9). Here, RCPL equalizer has only 3 nodes ($M = 3$) and the MLP equalizer has 2 hidden layers with 11 nodes in each layer. The RNN equalizer has the the same number of nodes as the RCPL filter. Its activation function is a hyperbolic tangent function. The RNN equalizer is modified such that it has the same structure as the RCPL filter, that is, no activation function is used in its output layer and no delay is employed between the most recent output of channel and the output of the equalizer. This modified RNN structure gives much better results than standard RNN structure given in [8]. The variance of noise is 0.01 in all the simulations. As seen in this example, performance of RCPL equalizer which has *piecewise linear* activation function is much more superior to that of the MLP equalizer and it exhibits comparable performance with that of the modified RNN equalizer which uses *hyperbolic tangent* activation function. We also compare the performance of RCPL equalizer by using different learning rates in Figure 6. Th simulation results shows that the learning rate a_1 which corresponds to the linear part of RCPL equalizer (i.e, the filter with index 0 in Figure 1) plays an important role in the learning processing in that it controls the rate of convergence. The choice of learning overall rate a_2 which corresponds to the nonlinear part (i.e, the filters with indices $1, \cdots, M$ in Figure 1) is more flexible than a_1 since the choice of order M is more important than that of the learning rate for this part.

REFERENCES

[1] T. Adalı and X. Liu, "Recurrent canonical piecewise linear network for nonlinear filtering and its application to blind equalization," accepted by *Signal Processing* (invited).

[2] C. P. Callender, C. F. N. Cowan, and S. Theodoridis, "The piecewise linear adaptive filter for channel equalisation in digital communications," in *Signal Processing VI: Theories and Applications,* 1992, pp. 143-146.

[3] S. Chen, S. McLaughlin, and B. Mulgrew, "Complex-valued radial basis function network, Part1: Network architecture and learning algorithms," *Signal Processing,* 35, pp. 19-31, 1994.

[4] L. O. Chua and A. Deng, "Canonical piecewise-Linear networks," *IEEE Trans. Circuits Syst.*, vol. 35, No. 1, pp. 101–111 , Jan. 1988.

[5] D. Donoho, "On minimum entropy deconvolution," in *Applied time series analysis II,* ed. D. Findley, Academic Press, New York, pp. 556-608.

[6] G. J. Gibson, S. Siu, and C. F. N. Cowan, "The application of nonlinear structures to the reconstruction of binary signals," in *IEEE Trans. Signal Processing,* vol. 39, no. 8, pp. 1877-1884, Aug. 1991.

[7] D. N. Godard, "Self-recovering equalization and carrier tracking in two-dimensional data communication systems," *IEEE Trans. Commun.,* vol. COM-28, pp. pp. 1867-1875. Nov. 1980.

[8] G. Kechriotis, E. Zervas, and E. S. Manolakos, "Using recurrent neural networks for adaptive communication channel equalization," *IEEE Trans. on Neural Networks,* vol. 5, no. 2, pp. 267-278, March 1994.

[9] J.N. Lin and R. Unbehauen,"Adaptive nonlinear digital filter with canonical piecewise-linear structure," in *IEEE Trans. Circuits Syst.,* vol.37, no.3, pp. 347-353.

[10] X. Liu and T. Adalı, "Recurrent canonical piecewise linear network and its application to adaptive equalization," in *Proc. Int. Conf. on Neural Networks* (Washington, DC), June 1996.

[11] X. Liu and T. Adalı, "Recurrent canonical piecewise linear network for blind equalization," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* (ICASSP), Munich, Germany, April, 1997.

[12] V. J. Mathews, "Adaptive polynomial filters," *IEEE Trans. Signal Processing Mag.,* pp. 10-26, July 1991.

[13] N. B. O. L. Pettit and P. E. Wellstead, "Analyzing piecewise linear dynamic systems," *IEEE Control Systems magazine,* vol. 8, pp. 43-50, 1995.

[14] W. Ruge, "Analytical framework for gain scheduling, " *IEEE Control System Magazine,*, vol 11, No. 1, pp. 74-84, 1991.

[15] S. Vembu, S. Verdu, R. A. Kennedy, and W. Sethares, "Convex cost functions in blind equalization," *IEEE Trans. Signal Processing,* vol. 42, No. 8, pp. 1952-1959, Aug. 1994.

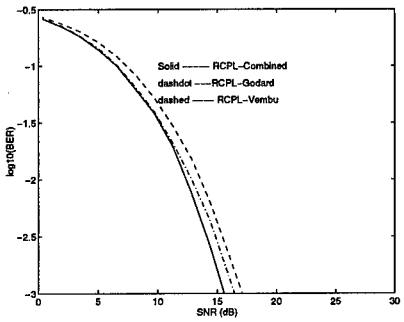


Figure 1: BER comparison

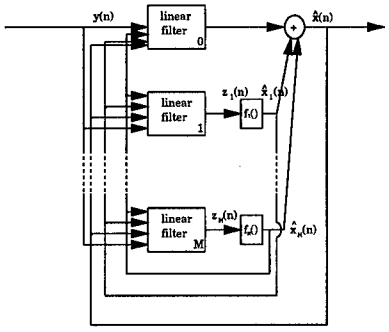


Figure 2: A simple RCPL network

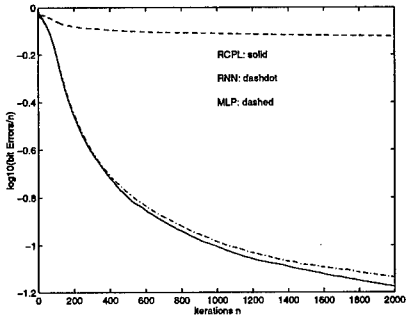


Figure 3: 16-PAM equalization

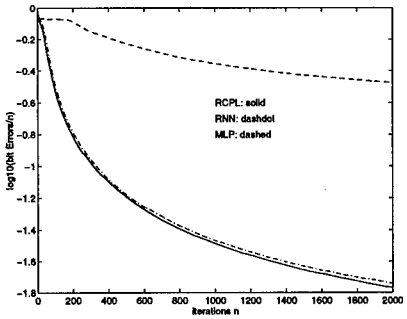


Figure 4: 8-PAM equalization

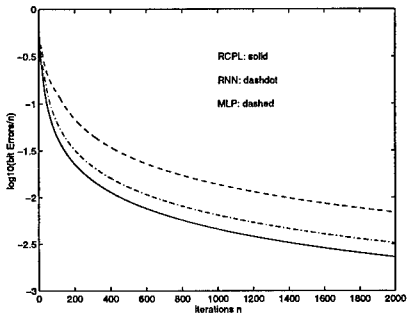


Figure 5: 2-PAM equalization

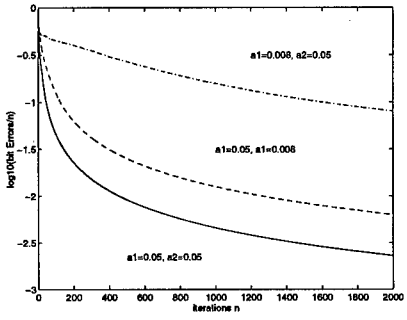


Figure 6: Learning rate comparison

BLIND SOURCE SEPARATION AND DECONVOLUTION BY DYNAMIC COMPONENT ANALYSIS

H. Attias* **and C.E. Schreiner†**

Sloan Center for Theoretical Neurobiology
University of California, San Francisco, CA 94143-0444

Abstract

We derive new unsupervised learning rules for blind separation of mixed and convolved sources. These rules are nonlinear in the signals and thus exploit high-order spatiotemporal statistics to achieve separation. The derivation is based on a global optimization formulation of the separation problem, yielding a stable algorithm. Different rules are obtained from frequency- and time-domain optimization. We illustrate the performance of this method by successfully separating convolutive mixtures of speech signals.

1 INTRODUCTION

In the problem of linear square blind separation [1], one considers L independent signal sources $x_i(t)$ (e.g., different speakers in a room) and L sensors $y_i(t)$ (e.g., microphones at several locations). Each sensor receives a mixture of the source signals. The task is to recover the original sources from the observed sensor signals. The separation is termed blind because it must be achieved without any information about the sources, apart from their statistical independence.

*Corresponding author. E-mail: hagai@phy.ucsf.edu. Phone: (415)476-1576. Fax: (415)502-4848.

†E-mail: chris@phy.ucsf.edu. Phone: (415)476-2591. Fax: (415)502-4848.

0-7803-4256-9/97/$10.00

Blind separation algorithms can have many applications in areas invloving processing of multi-sensor signals, such as speech enhancement (the 'cocktail party' problem) and the analysis and interpretation of biomedical signals (e.g., EKG, EEG [8]). Most of the separation methods that have been proposed aim at a simplified version of the problem where the mixing process is linear and instantaneous (memoryless). In that case we seek a separating transformation g_{ij} that, when applied to the sensor signals $y_i(t)$, will recover the sources, possibly scaled and permuted: $\hat{x}_i(t) = \sum_j g_{ij} y_j(t)$. In particular, independent component analysis (ICA) algorithms [2-7] can identify g_{ij} fast and efficiently in many cases.

However, the mixing in realistic situations is not memoryless, due to multipath propagation and the impulse response of the medium and of the sensors. The resulting 'convolutive' mixtures cannot be separated by ICA methods. In this paper we present a novel unsupervised learning algorithm for blind separation of linear, time-invariant mixtures with memory, termed *dynamic component analysis* (DCA). The separation in this case requires a transformation with a dynamic impulse response $g_{ij}(t)$ (a matrix of filters),

$$\hat{x}_i(t) = \sum_{j=1}^{L} \int_0^\infty dt' g_{ij}(t') y_j(t - t') \,, \tag{1}$$

where $\hat{x}_i(t)$ are the recovered source signals. More generally, the signals $y_i(t)$ may be taken from any temporal multi-sensor data set; the new signals $\hat{x}_i(t)$ are termed the *dynamic components* (DC) of those data.

Like the original sources, the DC's are characterized by their statistical independence, and consequently by the property that their joint moments factorize. In the time domain, this implies $\langle \hat{x}_i(t)^m \hat{x}_j(t+\tau)^n \rangle = \langle \hat{x}_i(t)^m \rangle \langle \hat{x}_j(t+\tau)^n \rangle$, for $i \neq j$ and all orders m, n at any time lag τ; the average is taken over time t. Note that in contrast, the *independent components* found by ICA algorithms satisfy this property only for $\tau = 0$.

In order to find the separating transformation $g_{ij}(t)$, one could impose the joint moment factorization as a condition on the resulting signals $\hat{x}_i(t)$. Rather than imposing it explicitly, which can practically be done only for low-order moments [14], an effective way to impose this condition implicitly and to all orders is to formulate the separation task as an optimization problem via the use of a latent-variable model [10]. Specifically, we construct a model for the joint distribution of the sensor signals over N-point time blocks, $p_y[\mathbf{y}(t_0), ..., \mathbf{y}(t_{N-1})]$, parametrized by the separating filter matrix

$\mathbf{g}(t_0), ..., \mathbf{g}(t_{M-1})$. Next, we define the 'distance' between our model sensor distribution and the observed distribution using the Kullback-Leibler (KL) distance [11], an information theory-based measure for the distance between two distributions. The model parameters are then optimized to minimize this distance by the stochastic gradient descent method, yielding the DCA learning rules for $g_{ij}(t)$.

This global optimization formulation of the problem can be given in either the frequency domain or the time domain. Section 2 presents the frequency-domain formulation and the associated learning rules, whereas the time-domain version is given in Section 3. The performance of DCA is illustrated in Section 4 by successfully separating convolutive mixtures of speech signals.

Notation: we work in discrete time t_n. Lower-case symbols are used for time-domain quantities and upper-case symbols for their frequency-domain counterparts. We use subscripts to refer to discrete times and frequencies, e.g., $x_n = x(t_n)$ and $X_k = X(\omega_k)$. Vectors and matrices are boldfaced.

2 FREQUENCY-DOMAIN OPTIMIZATION

Let $\mathbf{x}_n$ be the L-dimensional model source vector, whose elements $x_{i,n} = x_i(t_n)$ are the source activities at time t_n; these are the latent variables. Let $\mathbf{y}_n$ be the L-dimensional model sensor vector. We work with N-point time blocks $\{t_n\}, n = 0, ..., N-1$. The two are related by

$$\mathbf{x}_n = \sum_{m=0}^{M-1} \mathbf{g}_m \mathbf{y}_{n-m} \ , \qquad \mathbf{X}_k = \mathbf{G}_k \mathbf{Y}_k \ , \tag{2}$$

where the separating transformation $\mathbf{g}_m$ is a matrix of filters of length $M \ll N$, and $\mathbf{G}_k = \mathbf{G}(\omega_k)$ is its N-point DFT. We focus first of the frequency-domain formulation (r.h.s. of (2)) where the separation problem factorizes.

To construct a model sensor distribution $p_Y(\{\mathbf{Y}_k\})$ we must start with a model source distribution $p_X(\{\mathbf{X}_k\})$. We use a factorial frequency-domain model,

$$p_X(\{\mathbf{X}_k\}) = \prod_{i=1}^{L} \prod_{k=1}^{N/2-1} P_{i,k}(X_{i,k}) \ , \tag{3}$$

where $P_{i,k}$ is the joint distribution of $\mathrm{Re}X_{i,k}, \mathrm{Im}X_{i,k}$. From (2) we obtain $p_Y = \prod_k \det(\mathbf{G}_k \mathbf{G}_k^\dagger) p_X$, which depends on the separating parameters $\mathbf{g}_m$ and the parameters used to describe $P_{i,k}$ (see below).

Denoting the observed sensor signals by $\tilde{\mathbf{Y}}_k$, we now define a distance measure D between their joint distribution $p_{\tilde{Y}}$ and our model distribution p_Y. For this purpose we adopt the KL distance function [11], which can be shown to satisfy $D(p_{\tilde{Y}}, p_Y) = -H_{\tilde{Y}} - \langle \log p_Y \rangle_{\tilde{Y}}$; the second term on the r.h.s. is evaluated by averaging $\log p_Y(\tilde{Y})$ using the observed distribution $p_{\tilde{Y}}$. Since $H_{\tilde{Y}}$, the entropy of the observed signals, is independent of the mixing model parameters, minimizing D is equivalent to maximizing the log-likelihood of the data, $\langle \log p_Y \rangle_{\tilde{Y}}$, with respect to $\mathbf{g}_m$. It follows that

$$D(p_{\tilde{Y}}, p_Y) = -\frac{1}{N} \sum_{k=1}^{N/2-1} \left(\log \det \mathbf{G}_k \mathbf{G}_k^\dagger + \sum_{i=1}^{L} \log P_{i,k} \right) , \tag{4}$$

after dropping the average sign and terms independent of $\mathbf{g}_m$.

Before deriving the learning rules we make a few simplifications in the model (3) by omitting the frequency dependence of $P_{i,k}$ and using the same parametrized functional form for all sources. In addition, we restrict $P_{i,k}(X_{i,k})$ to depend only on the squared amplitude $\mid X_{i,k} \mid^2$. These simplifications are made for convenience, but a more complicated parametrization can be used in situations where the actual source distribution depends non-trivially on the frequency or phase. Note that our model sources are white, in anticipation of the whitening effect discussed below. Hence $P_{i,k}(X_{i,k}) = P(\mid X_{i,k} \mid^2; \xi_i)$, where ξ_i is a vector of parameters for source i. For instance, P may be a mixture of Gaussian distributions whose means, variances and weights are contained in ξ_i.

The frequency-domain DCA learning rules for the separating filters $\mathbf{g}_m$ and the source distribution parameters ξ_i are now obtained using a stochastic gradient descent minimization of the KL distance (4):

$$\begin{aligned} \delta \mathbf{G}_k &= \epsilon \left[\mathbf{I} - \Phi(\mathbf{X}_k) \mathbf{X}_k^\dagger \right] \mathbf{G}_k , \\ \delta \xi_i &= \epsilon \frac{1}{N} \sum_{k=1}^{N/2-1} \frac{\partial}{\partial \xi_i} \log P(\mid X_{i,k} \mid^2; \xi_i) , \end{aligned} \tag{5}$$

where $\delta \mathbf{g}_m$ are obtained from $\delta \mathbf{G}_k$ by inverse DFT for $0 \leq m \leq M-1$ and are set to zero for $m \geq M$. The vector $\Phi(\mathbf{X}_k)$ above is related to the model source distribution by

$$\Phi(X_{i,k}; \xi_i) = -X_{i,k} \frac{\partial}{\partial a} \log P(a = \mid X_{i,k} \mid^2; \xi_i) . \tag{6}$$

The learning rate is set by ϵ.

We point out that to derive the rule for $\mathbf{g}_m$ we used $\delta\mathbf{g}_m = -\epsilon\partial D/\partial\mathbf{g}_m$, but since the resulting $\delta\mathbf{G}_k$ required matrix inversion for each ω_k at each iteration, we multiplied it by the positive-definite matrix $\mathbf{G}_k^\dagger\mathbf{G}_k$ to get the less expensive rule (5). It can be shown [9] that this rule indeed decreases D at each iteration in the small-ϵ limit. Furthermore, it can be shown to satisfy the property of equivariance (see [6,7] for equivariant algorithms for instantaneous mixing), which guarantees uniform performance across all invertible mixing processes.

We emphasize the importance of using a time block that is sufficiently longer than our model filters. As is evident in the frequency-domain formulation (r.h.s. of (2) and $\delta\mathbf{G}_k$ rule of (5)), we are effectively solving N individual mixing problems, one at each ω_k, and risk recovering the sources with different ordering permutation at different frequencies, possibly reducing the separation quality. The key point here is that these N problems (or N increments $\delta\mathbf{G}_k$) are not independent, since the minimization of the distance function with respect to the M time-domain coefficients $\mathbf{g}_m$ couples them and solves them simultaneously. Consequently, to minimize the freedom of arbitrary permutations by exploiting this coupling we must choose $M \ll N$. Note that this difficulty does not reflect a limitation of any particular algorithm; rather, it is inherent to the convolutive mixing problem.

3 TIME-DOMAIN OPTIMIZATION

We now derive the learning rules for the separating filter matrix $\mathbf{g}_m$ starting from the time-domain description (2). For the model source distribution we use the factorial form

$$p_{\underset{\sim}{x}}(\{\mathbf{x}_m\}) = \prod_{i=1}^{L}\prod_{m=0}^{N-1} p_{i,m}(x_{i,m}) \,. \tag{7}$$

Using (7) together with (2), it is straightforward to derive the time-domain model sensor distribution p_y and its KL distance D to the observed distribution $p_{\tilde{y}}$:

$$D(p_{\tilde{y}}, p_y) = -\log\det\mathbf{g}_0 - \frac{1}{N}\sum_{m=0}^{N-1}\sum_{i=1}^{L}\log p_{i,m} \,. \tag{8}$$

As in the frequency-domain case, we simplify the model (7) by omitting the t_m-dependence (assuming stationary sources) and using the same functional form for all sources, parametrized by the vector ξ_i. Hence $p_{i,m}(x_{i,m}) =$

$p(x_{i,m};\xi_i)$. Stochastic gradient descent minimization of the KL distance (8) yields the time-domain DCA learning rules:

$$\begin{aligned}\delta\mathbf{G}_k &= \epsilon(\mathbf{g}_0^T)^{-1} - \epsilon\frac{1}{N}\Psi_k(\mathbf{x})\mathbf{Y}_k^\dagger \,, \\ \delta\xi_i &= \epsilon\frac{1}{N}\sum_{m=0}^{N-1}\frac{\partial}{\partial\xi_i}\log p(x_{i,m};\xi_i) \,, \end{aligned} \tag{9}$$

where $\delta\mathbf{g}_m$ is obtained from $\delta\mathbf{G}_k$ by inverse DFT. The vector $\Psi_k(\mathbf{x})$ above is the DFT of $\psi(\mathbf{x}_m)$: $\Psi_k(\mathbf{x}) = \sum_m e^{-i\omega_k m}\psi(\mathbf{x}_m)$. The latter is related to the model source distribution by the logarithmic derivative

$$\psi(x_{i,m};\xi_i) = -\frac{\partial}{\partial a}\log p(a = x_{i,m};\xi_i) \,. \tag{10}$$

The $\delta\mathbf{G}_k$ rule (9) is related to the rule derived in [12] (see also [13]) using information maximization considerations. It is not equivariant and is therefore not as efficient as (5).

4 SEPARATION OF SPEECH SIGNALS

We illustrate the performance of DCA by applying it to a convolutive mixture of speech signals. We mixed two 10sec-long signals, obtained from a commercial CD at the original sampling rate of 44.1KHz and down-sampled to $f_s = 4.41$KHz, by filters $h_{ij,n}$, whose impulse response is displayed in Figure 1. We then used the learning rules (5) and (9) to find the separating filters $g_{ij,n}$. The signals were processed in 512-point non-overlapping blocks, incrementing the separating filters after each block with $\epsilon = 0.01$.

We used an exponential form for the model source distribution, $p \propto e^{-\xi|x|}$, which approximates the distribution of the speech signals, as well as a large class of natural signals [15]. We also experimented with other distributions, such as the sigmoid-derivative form $p \propto e^{-\xi x}/(1+e^{-\xi x})^2$ used in [3]. Note that for the frequency-domain rule it is necessary to scale the variance by N or, alternatively, to modify the DFT definition by $\sqrt{N}$.

To demonstrate that separation has actually been accomplished, we present the convolution $(g \star h)_{ij,n}$ of the separating with the mixing filters in Figure 2. In the case of time-domain separation (solid line) the non-diagonal filters $(g \star h)_{12,n}$ and $(g \star h)_{21,n}$ are strongly attenuated compared to the diagonal ones, whereas in the case of frequency-domain separation (dashed line) the opposite is true. Thus separation has been achieved by both learning rules, followed by an order permutation for the latter.

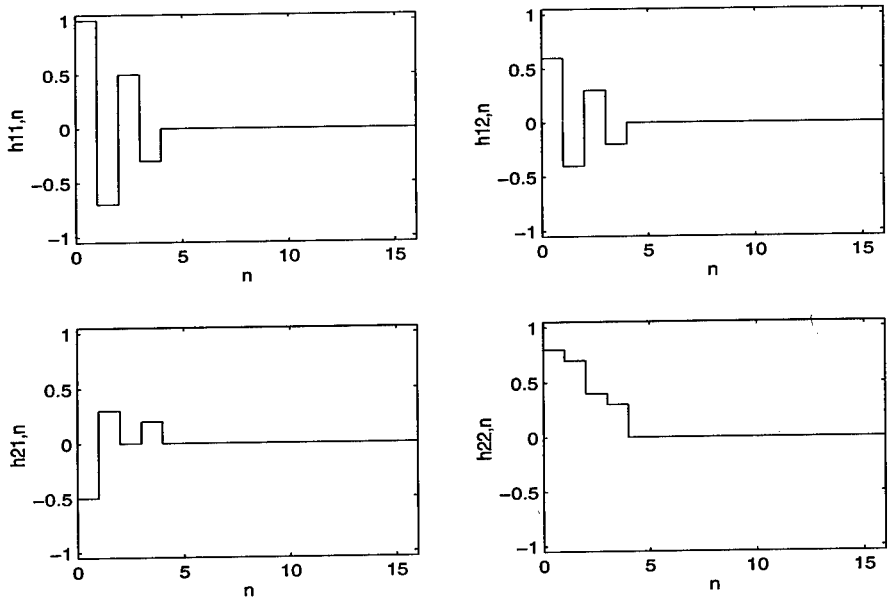


Figure 1: Impulse response of the mixing filters $h_{ij,n}$.

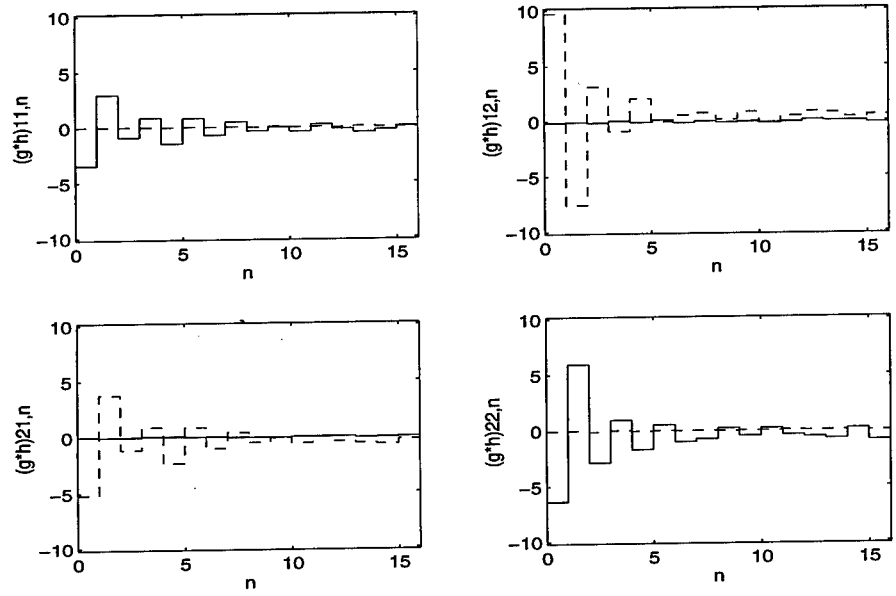


Figure 2: Convolution of the separating filters $g_{ij,n}$, obtained from time-domain (solid line) and frequency-domain (dashed line) optimization, with the mixing filters $h_{ij,n}$.

Note that the separation has modified the source power spectra, since for the diagonal solid-line filters $(g \star h)_{ii,n} \neq 0$ for $n > 0$, and similarly for the non-diagonal dashed-line filters. This whitening effect results from the fact that a general convolutive mixing situation is defined only to within an arbitrary permutation and filtering of the sources, just as an instantaneous mixing situation is defined to within source permutation and scaling. This ambiguity is evident in the frequency-domain form of (2): indeed, the distinction between the power spectrum of the sensor signals $\langle \mid Y_{j,k} \mid^2 \rangle$ and the separating transformation $G_{ij,k}$ cannot be made without prior information about the sources or the mixing situation.

5 CONCLUSION

In this paper we presented an optimization formulation of the problem of convolutive mixing, which allowed us to derive the DCA rules for learning the separating filter matrix from the observed mixtures in an unsupervised fashion. This formulation is advantageous since it results in a stable algorithm and facilitates a systematic derivation of time- and frequency-domain learning rules.

The DCA learning rules are non-linear in the signals and require processing in N-point time blocks. Consequently, this algorithm exploits high-order temporal and inter-sensor statistics to achieve separation. The rules presented here whiten the recovered sources; in [9] we show a way to avoid this by learning the source spectra as additional model parameters. Consequently, we must impose appropriate constraints on the mixing model to avoid the ambiguity mentioned above. To do that we derive learning rules for the mixing transformation. Those rules, however, do not satisfy the equivariant property, in contrast with the frequency-domain rules for the separating transformation derived in the present paper; nevertheless, in addition to facilitating the imposition of constraints, learning the mixing filters has the advantage of reducing model complexity since they are usually shorter than the separating filters.

Algorithms that solve the problem of blind source separation address, in fact, the more general need for an efficient tool for statistical analysis of temporal multi-variable data sets. We are currently using DCA to perform source analysis of auditory evoked potentials in magnetoencephalogram (MEG) multichannel recordings [16], where this technique is capable of separating the

contributions to the sensor signals from different sources of neural activity that respond simultaneously to the stimulus.

References

[1] C. Jutten and J. Herault, "Blind separation of sources, Part I: An adaptive algorithm based on neuromimetic architecture", *Signal Processing*, vol. 24, pp. 1-10, 1991.

[2] P. Comon, P, "Independent component analysis: a new concept?", *Signal Processing*, vol. 36, pp. 287-314, 1994.

[3] A.J. Bell and T.J. Sejnowski, "An information-maximization approach to blind separation and blind deconvolution", *Neural Computation*, vol. 7, pp. 1129-1159, 1995.

[4] B.A. Pearlmutter and L.C. Parra, "A context-sensitive generalization of ICA", in *Proceedings of the International Conference on Neural Information Processing*, Springer, 1996.

[5] D.T. Pham, "Blind separation of instantaneous mixture of sources via an independent component analysis", *IEEE Transactions on Signal Processing*, vol. 44, pp. 2768-2779, 1996.

[6] J.-F. Cardoso and B.H. Laheld, "Equivariant adaptive source separation", *IEEE Transactions on Signal Processing*, 1997, in press.

[7] S. Amari, A. Cichocki, and H.H. Yang, "A new learning algorithm for blind signal separation", in *Advances in Neural Information Processing Systems 8*, MIT press, Cambridge, MA, 1996.

[8] S. Makeig, A.J. Bell, T.-P. Jung, and T.J. Sejnowski, "Independent component analysis of electroencephalographic data", in *Advances in Neural Information Processing Systems 8*, MIT Press, Cambridge, MA, 1996.

[9] H. Attias and C.E. Schreiner, "Blind source separation and deconvolution: the dynamic component analysis algorithm", *Neural Computation*, submitted.

[10] B.S. Everitt, *An Introduction to Latent Variable Models*, Chapman and Hall, London, 1984.

[11] T.M. Cover and J.A. Thomas, *Elements of Information Theory*, John Wiley, New York, 1991.

[12] K. Torkkola, "Blind separation of convolved sources based on information maximization", in *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing*, 1996.

[13] A.J. Bell, T.-W. Lee, and R. Lambert, "Blind separation of delayed and convolved sources", in *Advances in Neural Information Processing Systems*

9, 1997, in press.
[14] D. Yellin and E. Weinstein, "Criteria for multichannel signal separation", *IEEE Transactions on Signal Processing*, vol. 42, pp. 2158-2168, 1995.
[15] H. Attias and C.E. Schreiner, "Low-order temporal statistics of natural sounds", in *Advances in Neural Information Processing Systems 9*, 1997, in press.
[16] D. Poeppel, H. Attias, H.A. Rowley, and C.E. Schreiner, "Dynamic component analysis of auditory evoked neuromagnetic fields", in *Society for Neuroscience Abstracts*, vol. 23, 1997.

NEURAL DUAL EXTENDED KALMAN FILTERING: APPLICATIONS IN SPEECH ENHANCEMENT AND MONAURAL BLIND SIGNAL SEPARATION

Eric A. Wan and Alex T. Nelson

Department of Electrical Engineering, Oregon Graduate Institute,
P.O. Box 91000 Portland, OR 97291

Abstract

The removal of noise from speech signals has applications ranging from speech enhancement for cellular communications, to front ends for speech recognition systems. A nonlinear time-domain method called *dual extended Kalman filtering* (DEKF) is presented for removing nonstationary and colored noise from speech. We further generalize the algorithm to perform the blind separation of two speech signals from a single recording.

INTRODUCTION

Traditional approaches to noise removal in speech involve spectral techniques, which frequently result in audible distortion of the signal. Recent time-domain nonlinear filtering methods utilize data sets where the clean speech is available as a target signal to train a neural network. Such methods are often effective within the training set, but tend to generalize poorly for actual sources with varying signal and noise levels. Furthermore, the network models in these methods do not fully take into account the nonstationary nature of speech. In the approach presented here, we assume the availability of only the noisy signal. Effectively, a *sequence* of neural networks is trained on the specific noisy speech signal of interest, resulting in a nonstationary model which can be used to remove noise from the given signal.

A noisy speech signal $y(k)$ can be accurately modeled as a nonlinear autoregression with both process and additive observation noise:

$$x(k) = f(x(k-1), \ldots x(k-M), \mathbf{w}) + v(k) \quad (1)$$

$$y(k) = x(k) + n(k), \quad (2)$$

where $x(k)$ corresponds to the true underlying speech signal driven by process noise $v(k)$, and $f(\cdot)$ is a nonlinear function of past values of $x(k)$ parameterized by $\mathbf{w}$. The speech is only assumed to be stationary over short segments, with each segment having a different model. The available observation is $y(k)$, which contains additive noise $n(k)$. The optimal *estimator* given the noisy observations $\mathbf{y}(k) = \{y(k), y(k-1), \cdots y(0)\}$ is $E[x(k)|\mathbf{y}(k)]$. The most direct way to estimate this would be to train on a set of clean data in which the true $x(k)$ may be used as the target to a neural

0-7803-4256-9/97/$10.00

network. Our assumption, however, is that the clean speech is never available; the goal is to estimate $x(k)$ itself from the noisy measurements $y(k)$ alone.

In order to solve this problem, we assume that $f(\cdot,\cdot)$ is in the class of feedforward neural network models, and compute the dual estimation of both states $\hat{x}$ and weights $\hat{\mathbf{w}}$ based on a Kalman filtering approach. In this paper we provide a basic description of the algorithm, followed by a discussion of experimental results.

DUAL EXTENDED KALMAN FILTERING

By posing the dual estimation problem in a state-space framework, we can use Kalman filtering methods to perform the estimation in an efficient, recursive manner. At each time point, the Kalman filter provides an optimal estimation by combining a prior prediction with a new observation. Connor *et al.*[4], proposed using an extended Kalman filter with a neural network to perform state estimation alone. Puskorious and Feldkamp [13] and others have posed the weight estimation in a state-space framework to allow for efficient Kalman training of a neural network. In prior work, we extended these ideas to include the dual Kalman estimation of both states and weights for efficient maximum-likelihood optimization (in the context of robust nonlinear prediction, estimation, and smoothing) [15]. The work presented here develops these ideas in the context of speech processing.

A state-space formulation of (1) and (2) is as follows:

$$\mathbf{x}(k) = F[\mathbf{x}(k-1)] + Bv(k), \tag{3}$$

$$y(k) = C\mathbf{x}(k) + n(k), \tag{4}$$

where

$$\mathbf{x}(k) = \begin{bmatrix} x(k) \\ x(k-1) \\ \vdots \\ x(k-M+1) \end{bmatrix}, \quad F[\mathbf{x}(k)] = \begin{bmatrix} f(x(k),\ldots,x(k-M+1),\mathbf{w}) \\ x(k) \\ \vdots \\ x(k-M+2) \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}, \qquad B = C^T \tag{5}$$

If the model is linear, then $f(\mathbf{x}(k))$ takes the form $\mathbf{w}^T\mathbf{x}(k)$, and $F[\mathbf{x}(k)]$ can be written as $A\mathbf{x}(k)$, where A is in controllable canonical form. We initially assume the noise terms $v(k)$ and $n(k)$ are white with known variances σ_v^2 and σ_n^2, respectively. Methods for estimating the noise variances directly from the noisy data are described later in this paper.

Extended Kalman Filter - State Estimation

For a *linear* model with *known* parameters, the Kalman filter (KF) algorithm can be readily used to estimate the states [9]. At each time step, the filter computes the linear least squares estimate $\hat{x}(k)$ and prediction $\hat{x}^-(k)$, as well as their error covariances, $P_X(k)$ and $P_{\tilde{X}}(k)$. In the linear case with Gaussian statistics, the estimates are the

minimum mean square estimates. With no prior information on x, they reduce to the maximum likelihood estimates.

When the model is nonlinear, the KF cannot be applied directly, but requires a linearization of the nonlinear model at the each time step. The resulting algorithm is called the extended Kalman filter (EKF), and effectively approximates the nonlinear function with a time-varying linear one. The EKF algorithm is as follows:

$$\hat{\mathbf{x}}^-(k) = F[\hat{\mathbf{x}}(k-1), \hat{\mathbf{w}}(k-1)] \tag{6}$$

$$P_{\hat{\mathbf{x}}}^-(k) = AP_{\hat{\mathbf{x}}}(k-1)A^T + B\sigma_v^2 B^T, \quad \text{where} \quad A = \left.\frac{\partial F[\hat{\mathbf{x}}, \hat{\mathbf{w}}]}{\partial \hat{\mathbf{x}}}\right|_{\hat{\mathbf{x}}(k-1)} \tag{7}$$

$$K(k) = P_{\hat{\mathbf{x}}}^-(k)C^T(CP_{\hat{\mathbf{x}}}^-(k)C^T + \sigma_n^2)^{-1} \tag{8}$$

$$P_{\hat{\mathbf{x}}}(k) = (I - K(k)C)P_{\hat{\mathbf{x}}}^-(k) \tag{9}$$

$$\hat{\mathbf{x}}(k) = \hat{\mathbf{x}}^-(k) + K(k)(y(k) - C\hat{\mathbf{x}}^-(k)). \tag{10}$$

Dual Extended Kalman Filter - Weight Estimation

Because the model for the speech is not known, the standard EKF algorithm cannot be applied directly. We approach this problem by constructing a separate state-space formulation for the underlying weights as follows:

$$\mathbf{w}(k) = \mathbf{w}(k-1) \tag{11}$$

$$y(k) = f(\mathbf{x}(k-1), \mathbf{w}(k)) + v(k) + n(k), \tag{12}$$

where the state transition is simply an identity matrix, and the neural network $f(\mathbf{x}(k-1), \mathbf{w}(k))$ plays the role of a time-varying nonlinear observation on $\mathbf{w}$. These state-space equations for the weights allow us to estimate them with a second EKF.

$$\hat{\mathbf{w}}^-(k) = \hat{\mathbf{w}}(k-1) \tag{13}$$

$$P_{\hat{\mathbf{w}}}^-(k) = P_{\hat{\mathbf{w}}}(k-1) \tag{14}$$

$$K_{\hat{\mathbf{w}}}(k) = P_{\hat{\mathbf{w}}}^-(k)H(k)^T(H(k)P_{\hat{\mathbf{w}}}^-(k)H(k)^T + \sigma_n^2 + \sigma_v^2)^{-1} \tag{15}$$

$$P_{\hat{\mathbf{w}}}(k) = (I - K_{\hat{\mathbf{w}}}(k)H(k))P_{\hat{\mathbf{w}}}^-(k) \tag{16}$$

$$\hat{\mathbf{w}}(k) = \hat{\mathbf{w}}^-(k) + K_{\hat{\mathbf{w}}}(k)(y(k) - CF(\hat{\mathbf{x}}(k-1), \hat{\mathbf{w}}^-(k))) \quad , \tag{17}$$

$$\text{where} \quad H(k) = \left.\frac{C\partial F[\hat{\mathbf{x}}, \hat{\mathbf{w}}]}{\partial \hat{\mathbf{w}}}\right|_{\hat{\mathbf{w}}(k-1)} . \tag{18}$$

The linearization in (18) can be computed as a dynamic derivative [16] to account for the recurrent nature of the state-estimation filter, including the dependence of the Kalman gain $K(k)$ on the weights. The calculation of these derivatives is computationally expensive, and can be avoided by ignoring the dependence of $\hat{\mathbf{x}}(k)$ on $\hat{\mathbf{w}}$.[1] This approximation was used to produce the results in this paper. The use of the full derivatives is currently being investigated by the authors.

[1] This is equivalent to a single-step of backpropagation through time [16].

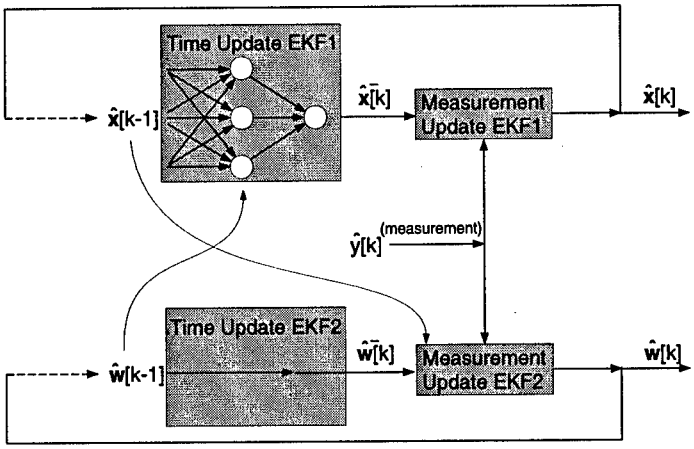


Figure 1: The Dual Extend Kalman Filter (DEKF). EKF1 and EKF2 represent the filters for the states and the weights, respectively.

We now have EKFs for estimating both the states $\mathbf{x}$ and the weights $\mathbf{w}$, resulting in a pair of *dual extended Kalman filters* (DEKF) run in parallel (see Figure 1). At each time step, the current estimate of $\mathbf{x}$ is used by the weight filter, and the current estimate of $\mathbf{w}$ is used by the state filter. For finite data sets, the algorithm is run iteratively over the data until the weights converge.

This approach to dual estimation is related to work done by Nelson [12] in the linear case, and to Matthews' neural approach [11] to the recursive prediction error algorithm [7][2]. In the speech literature, the method is most closely related to Lim and Oppenheim's approach to fitting LPC models to degraded speech [10]. It also relates to Ephraim's model-based approach [6], but uses nonlinear estimation to fit the given data instead of using a fixed number of prespecified linear models.

Nonstationary White Noise Experiment

The result of applying the DEKF to a speech signal corrupted with simulated nonstationary bursting noise is shown in Figure 2. The method was applied to successive 64ms (512 point) windows of the signal, with a new window starting every 8ms (64 points).[3] Feedforward networks with 10 inputs, 4 hidden units, and 1 output were used. Weights typically converged in less than 20 epochs. The results in the figure were computed assuming both σ_v^2 and σ_n^2 were known. The average SNR is improved by 9.94 dB, with little resultant distortion. We also ran the experiment when σ_n^2 and σ_v^2 were estimated using only the noisy signal. This also produced impressive results,

[2] An alternative approach is to concatenate both $\mathbf{w}$ and $\mathbf{x}$ into a *joint* state vector, and apply the EKF to the resulting nonlinear state equations (see [7] for the linear case, [17] for application to recurrent neural networks). This algorithm, however, has been known to have convergence problems.

[3] A normalized Hamming window was used to emphasizes data in the center of the window, and deemphasize data in the periphery. The standard EKF equations are also modified to reflect this windowing in the weight estimation.

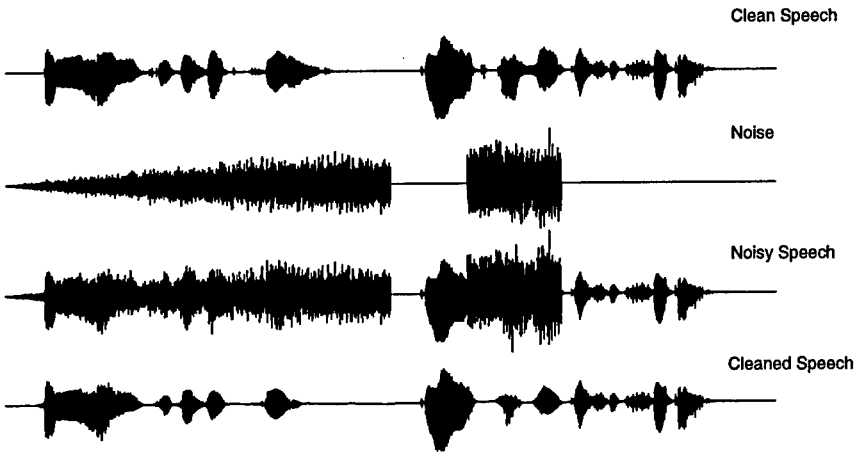


Figure 2: Cleaning Noisy Speech With The DEKF. The speech data was approximately 33,000 points (4 seconds) long. Nonstationary white noise was generated artificially and added to the speech to create the noisy signal y.

with an SNR improvement of 8.50 dB. In comparison, available "state-of-the-art" techniques of *spectral subtraction* [3] and *adaptive RASTA* processing [1] achieve SNR improvements of only .65 and 1.26 dB, respectively.

Colored Noise

For most real-world speech applications, we cannot assume the noise is white. For colored noise, the state-space equations 3 and 4 need to be adjusted before Kalman filtering techniques can be employed. Specifically, the measurement noise process is given its own state-space equations,

$$\mathbf{n}(k) = A_n \mathbf{n}(k-1) + B_n v_n(k) \quad (19)$$

$$n(k) = C_n \mathbf{n}(k), \quad (20)$$

where $\mathbf{n}(k)$ is a vector of lagged values of $n(k)$, $v_n(k)$ is white noise, A_n is a simple state transition matrix in controllable canonical form, and B_n and C_n are of the same form as B and C given in (3) and (4). Note that this is equivalent to an autoregressive model of the colored noise, which may be fit from a small section of signal where the speech is not present.

With this formulation for the colored noise, it is straightforward to augment both the state $\mathbf{x}(k)$ and the weight $\mathbf{w}(k)$ with $\mathbf{n}(k)$, and write down combined state equations. Specifically, (3) and (4) are replaced by:

$$\begin{bmatrix} \mathbf{x}(k) \\ \mathbf{n}(k) \end{bmatrix} = \begin{bmatrix} F[\mathbf{x}(k-1)] \\ A_n \mathbf{n}(k-1) \end{bmatrix} + \begin{bmatrix} B & 0 \\ 0 & B_n \end{bmatrix} \begin{bmatrix} v(k) \\ v_n(k) \end{bmatrix}, \quad (21)$$

$$y(k) = [C \;\; C_n] \begin{bmatrix} \mathbf{x}(k) \\ \mathbf{n}(k) \end{bmatrix}, \quad (22)$$

and (11) and (12) are replaced by:

$$\begin{bmatrix} \mathbf{w}(k) \\ \mathbf{n}(k) \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & A_n \end{bmatrix} \begin{bmatrix} \mathbf{w}(k-1) \\ \mathbf{n}(k-1) \end{bmatrix} + \begin{bmatrix} 0 \\ B_n \end{bmatrix} v_n(k), \quad (23)$$

$$y(k) = f(\mathbf{x}(k-1), \mathbf{w}(k)) + C_n \mathbf{n}(k) + v(k). \quad (24)$$

The noise processes in these state-equations are now white, and the DEKF algorithm can be used to estimate the signal. Note, the colored noise explicitly affects not only the state estimation, but also the weight estimation.

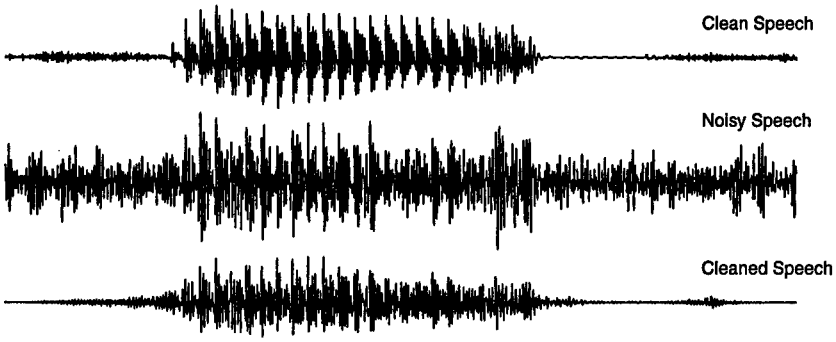


Figure 3: Removing Colored Noise With The DEKF. The speech data is 3,500 points long. An actual recording of stationary colored noise was added to the speech to create the noisy signal y.

An actual recording of cellular phone noise was added to a speech signal to produce the data shown in Figure 3. The noise was modeled as an AR(10) process to determine A_n and $\sigma^2_{v_n}$ using a segment of the data (512 points) where no speech is present. The results in the figure reflect assumed knowledge of σ^2_v, and showed an average SNR improvement of 5.77 dB. When σ^2_v is estimated, the SNR improvement is 5.71 dB. In this case, spectral subtraction and adaptive RASTA processing produced comparable SNR improvements of 4.87 dB and 5.27 dB, respectively.

MONAURAL BLIND SIGNAL SEPARATION

If the additive noise is colored and highly nonstationary, the distinction between what is signal and what is noise becomes somewhat arbitrary. For this reason, we consider the observation $y(k) = x(k) + n(k)$ to represent the addition of two signals $y(k) = x_1(k) + x_2(k)$. We simply treat the noise itself as an additional signal that must be estimated. This is a form of blind signal separation, in which, for example, the signals result from the mixing of speakers. The problem differs from recent methods in the literature [2] for blind signal separation in which M signals must be separated from M observations by learning a fixed "inverse weighting" matrix. Instead, we are interested in separating two or more signals from a *single* (monaural) observation.

Previous work on monaural signal separation has primarily been based on harmonic

selection and pitch tracking in the frequency domain [5]. In contrast, we estimate each signal by learning a set of short-term models which best separate the signals. Extension of the DEKF framework is straightforward. Specifically, we formulate state equations containing $\mathbf{x}_1(k)$ and $\mathbf{x}_2(k)$

$$\begin{bmatrix} \mathbf{x}_1(k) \\ \mathbf{x}_2(k) \end{bmatrix} = \begin{bmatrix} F_1[\mathbf{x}_1(k-1)] \\ F_2[\mathbf{x}_2(k-1)] \end{bmatrix} + \begin{bmatrix} B_1 & 0 \\ 0 & B_2 \end{bmatrix} \begin{bmatrix} v_1(k) \\ v_2(k) \end{bmatrix}, \quad (25)$$

$$y(k) = [C_1 \;\; C_2] \begin{bmatrix} \mathbf{x}_1(k) \\ \mathbf{x}_2(k) \end{bmatrix}. \quad (26)$$

These equations are analogous to the colored noise formulation (22) and (21), where $\mathbf{n}$ is replaced by $\mathbf{x}_2$ and has corresponding nonlinear model F_2 with parameters $\mathbf{w}_2$. To estimate the weight parameters $\mathbf{w}_1$ for the neural network F_1 associated with $\mathbf{x}_1$ we form the state equations

$$\begin{bmatrix} \mathbf{w}_1(k) \\ \mathbf{x}_2(k) \end{bmatrix} = \begin{bmatrix} \mathbf{w}_1(k-1) \\ F_2[\mathbf{x}_2(k-1])] \end{bmatrix} + \begin{bmatrix} 0 \\ B_2 \end{bmatrix} v_2(k), \quad (27)$$

$$y(k) = f_1(\mathbf{x}_1(k-1), \mathbf{w}_1(k)) + C_2\mathbf{x}_2(k) + v_1(k), \quad (28)$$

where the state $\mathbf{x}_2$ is included so that the associated noise process is white (compare to (23) and (24)). Finally, for estimating the second set of weight parameters $\mathbf{w}_2$ for F_2 associated with $\mathbf{x}_2$, we add a third set of state equations

$$\begin{bmatrix} \mathbf{w}_2(k) \\ \mathbf{x}_1(k) \end{bmatrix} = \begin{bmatrix} \mathbf{w}_2(k-1) \\ F_1[\mathbf{x}_1(k-1])] \end{bmatrix} + \begin{bmatrix} 0 \\ B_1 \end{bmatrix} v_1(k), \quad (29)$$

$$y(k) = f_2(\mathbf{x}_2(k-1), \mathbf{w}_2(k)) + C_1\mathbf{x}_1(k) + v_2(k). \quad (30)$$

It is straightforward to show that given a known (linear) model for each signal, both $x_1(k)$ and $x_2(k)$ are *observable* from the additive observation $y(k)$. However, showing that model parameters may be jointly *learned* from the observations alone is a much more difficult problem. Nevertheless, some simple preliminary experiments have been performed which indicate the potential of this approach. Figure 4 illustrates the blind separation of two segments of speech (male /s/ and female /eI/) which have been added together and then separated in this manner. While encouraging, the approach should be viewed as only a starting point to a model based framework for approaching blind signal separation problems.

ESTIMATING NOISE VARIANCES

In the implementation of the DEKF, it is assumed that the variances of $v(k)$ and $n(k)$ in (3) and (4) are known quantities. In practical applications, however, the noise variances must be estimated from the noisy data. We have investigated several approaches for doing this in the speech processing domain.

Additive Noise Statistics: Assuming stationarity of the additive noise, the noise variance σ_n^2 (or its full autocorrelation) may be estimated from segments of the data $y(k)$

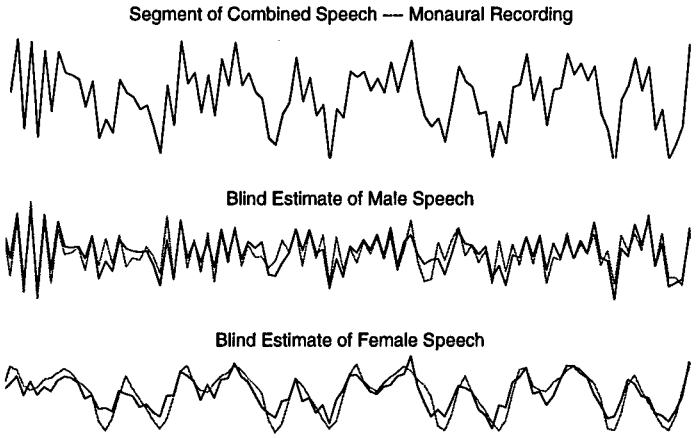


Figure 4: Blind Separation. Top figure shows the combined signal supplied to the algorithm. In the bottom two figures, the original speech is shown with a dashed line, and the estimated signal with a solid line. 100 data points are shown.

that do not contain speech. For slow variations in the noise statistics, Hirsch [8] has proposed an approach based on histograms of spectral magnitudes which does not require explicit segmentation of the data into speech and non-speech segments.

For rapidly changing noise (e.g., background chatter, wind noise, and artifacts introduced by automatic gain control) we are interested in *short-term* estimates of the noise statistics for each window of noisy speech data. An approach we have developed for nonstationary white noise sources re-estimates σ_n^2 as follows. First, we note that the optimal weights for the linear estimator

$$\hat{x}(k) = \sum_{i=0}^{M} w_i y(k-i) = \mathbf{w}^T \mathbf{y}(k), \tag{31}$$

may be expressed as

$$\hat{\mathbf{w}}^* = \hat{\mathbf{R}}_{yy}^{-1}(\hat{\mathbf{r}}_{yy} - \sigma_n^2 e_1), \tag{32}$$

where $\hat{\mathbf{R}}_{yy}$ is the sample autocorrelation of the noisy speech, and $e_1 = [1\ 0\ \cdots\ 0]$. Next, we choose $\hat{\sigma}_n^2$ in (32) such that $\hat{\mathbf{w}}$ leads to a *minimum variance* estimator. We have shown that this provides an upper bound on σ_n^2. Starting at this upper bound, $\hat{\sigma}_n^2$ is iteratively decreased until $\hat{w}_0 > \hat{w}_i\ \forall i \neq 0$, which forces the current observation to have the greatest influence the estimator output relative to other observations. A new $\hat{\sigma}_n^2$ is then re-estimated for each short-term window. This technique was used for the results given with the nonstationary white noise experiment.

Process Noise Variance: To estimate σ_v^2 (assuming an LPC model for the signal), Lim and Oppenheim [10] used an expression for the inverse Fourier transform of the

signal power (which is a function of σ_v^2). We have developed an alternative approach by noting that the process noise variance σ_v^2 can be estimated as the mean squared error of a linear AR predictor on the clean data $x(k)$[4]. Specifically,

$$\sigma_v^2 = \sigma_x^2 - \mathbf{p}_{xx}^T \mathbf{R}_{xx}^{-1} \mathbf{p}_{xx},$$

where $\mathbf{p}_{xx}$ is the cross-correlation between the lagged input vector $\mathbf{x}(k-1)$ and the current $x(k)$, and $\mathbf{R}_{xx}$ is the autocorrelation of the inputs. In our setup, only the noisy signal $y(k)$ with prediction residual $\sigma_x^2 + \sigma_n^2 - \mathbf{p}_{yy}^T \mathbf{R}_{yy}^{-1} \mathbf{p}_{yy}$ is available. We can approximate $\sigma_x^2 - \mathbf{p}_{xx}^T \mathbf{R}_{xx}^{-1} \mathbf{p}_{xx}$ using:

$$\hat{\sigma}_x^2 = \sigma_y^2 - \hat{\sigma}_n^2, \quad \hat{\mathbf{p}}_{xx} = \mathbf{p}_{yy} - \hat{\mathbf{p}}_{nn}, \quad \hat{\mathbf{R}}_{xx} = \mathbf{R}_{yy} - \hat{\mathbf{R}}_{nn}. \tag{33}$$

This results in the following estimate:

$$\hat{\sigma}_v^2 = (\hat{\sigma}_y^2 - \hat{\sigma}_n^2) - \hat{\mathbf{p}}_{xx}^T \hat{\mathbf{R}}_{xx}^{-1} \hat{\mathbf{p}}_{xx}. \tag{34}$$

Note that when $n(k)$ is white, the terms in (33) simplify because $\hat{\mathbf{p}}_{nn} = \hat{\sigma}_n^2 e_1$ and $\hat{\mathbf{R}}_{nn} = \hat{\sigma}_n^2 I$, where the additive noise variance is estimated as above.

While these "ad-hoc" methods were used in the experiments reported in this paper, estimating the noise variances remains a critical area for future work. Our current direction is to treat σ_v^2 and σ_n^2 as additional parameters which may be optimized within the Kalman and maximum-likelihood framework.

CONCLUSION AND FUTURE WORK

We have presented a DEKF algorithm with preliminary results on its application to speech enhancement in the presence of both nonstationary and colored noise. The approach performs significantly better than the current state-of-the-art on the reduction of nonstationary noise, and performs well on colored noise problems as well. In addition, the application of the approach to monaural blind signal separation was considered as a special case of the nonstationary colored noise problem.

Future work will include additional approaches to variance estimation, as well as the coupling of error statistics, windowing aspects, recurrent training implications, and forward-backward methods for *smoothing*. An additional aspect that is currently under consideration is the minimization of both prediction *and* estimation errors by the weight filter. While the current implementation minimizes only prediction error of the model, the full *errors in variables* cost function [14, 15] can be minimized by a two-observation form of the weight filter. This refinement will be discussed in a future paper.

Acknowledgments: This work was sponsored by the NSF under grant ECS-9410823, and by ARPA/AASERT Grant DAAH04-95-1-0485. We would like to thank C. Avendano for providing spectral subtraction and RASTA processing comparisons.

[4] This is exact, assuming the signal is generated by a linear autoregression.

REFERENCES

[1] C. Avendano, H. Hermansky, M. Vis, A. Bayya. Adaptive speech enhancement based on frequency-specific SNR estimation. *Proc. IVTTA 1996*, Basking Ridge, NJ, October 1996.

[2] A. Bell, T. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7, 1129-1159, 1995.

[3] S.F. Boll. Suppression of acoustic noise in speech using spectral subtraction. *IEEE ASSP-27*, pp. 113-120, April 1979.

[4] J. Connor, R. Martin, L. Atlas. Recurrent neural networks and robust time series prediction. *IEEE Tr. on Neural Networks*. March 1994.

[5] P.N. Denbigh, J. Zhao. Pitch extraction and separation of overlapping speech. *Speech Communication*, vol.11, pp.119-25. 1992.

[6] Y. Ephraim. Statistical model-based speech enhancement systems. *Proceedings of the IEEE*, Vol. 80, October 1992.

[7] G. Goodwin, K.S. Sin. *Adaptive Filtering Prediction and Control*. Prentice-Hall, Inc., Englewood Cliffs, NJ. 1994.

[8] H.G. Hirsch. Estimation of noise spectrum and its application to SNR-estimation and speech enhancement. *Technical Report TR-93-012, International Computer Science Institute*. 1993.

[9] F. Lewis. *Optimal Estimation*. John Wiley & Sons, Inc. New York, 1986.

[10] J. Lim, A. Oppenheim. All-pole modeling of degraded speech. *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 26, June 1978.

[11] M. B. Matthews and G. S. Moschytz. The identification of nonlinear discrete-time fading-memory systems using neural network models. *IEEE Transactions on Circuits and Systems-II*, 41(11):740–51, November 1994.

[12] L. Nelson, E. Stear. The simultaneous on-line estimation of parameters and states in linear systems. *IEEE Tr. on Automatic Control*. February 1976.

[13] G. Puskorious, L. Feldkamp. Neural control of nonlinear dynamic systems with Kalman filter trained recurrent networks. *IEEE Trn. on NN*, vol. 5, no. 2, 1994.

[14] G. Seber, C. Wild. *Nonlinear Regression*. John Wiley & Sons. 1989.

[15] E. Wan, A. Nelson. Dual Kalman filtering methods for nonlinear prediction, estimation, and smoothing. In *NIPS96 Proceedings*, 1997.

[16] P. Werbos. Backpropagation through time: what it does and how to do it. *Proc. IEEE, special issue on neural networks* **2**. 1550-1560. 1990.

[17] R. Williams. Training recurrent networks using extended Kalman filter. *International Joint Conference on Neural Networks*, Baltimore, vol. IV, 1992.

Bayesian Ying-Yang Learning Based ICA Models*

Lei Xu
Dept of Computer Science & Eng., Chinese Univ. of Hong Kong, HK

ABSTRACT

It has been shown that a particular case of the Bayesian Ying-Yang learning system and theory will reduce into a very general ICA framework. It not only includes the existing information-theoretic ICA approaches as particular examples, but also improve their performances, extend them to handle the cases that sensors are affected by noises and outliers and cases that the number of sensors is larger than the number of sources, and also be able to detect the correct number of sources. Algorithms are developed for implementing this ICA framework both in its general form and in its simplified versions for two important special cases, supported by some theoretical results and experimental demonstration.

1. Introduction

Recently, Independent Component Analysis (ICA) has received increasing attention with many applications [1]. Here, we consider a widely used formulation for the ICA problem. Given that there are n channels of unknown source signals $s = [s_1, \cdots, s_n]^T$ which are mutually independent with $Es = 0$. The observations from n sensors are given as $x = A_o s$ with A_o being an $n \times n$ unknown nonsingular mixing matrix. The objective is to find a so-called demixing matrix W such that

$$y = Wx = WA_o s = Vs, \quad V = WA_o = \Pi D \tag{1}$$

with Π being a permutation matrix and D being a non-singular diagonal matrix. That is, y recovers s up to unknown scales and a permutation of indices. Theoretically, we can get such an W as long as it makes $\{y_1, \cdots, y_n\}$ mutually independent when there is at most one signal in $[s_1, \cdots, s_n]^T$ is gaussian, which motivated the name [1,2]. A recent popular stream for the problem is called information-theoretic approach. One idea is the minimization of the mutual information (MMI) $J(W) = \int_y p(y) \ln p(y) / \prod_{i=1}^n p(y_i) dy$ because $J(W)$ is the minimum when $\{y_1, \cdots, y_n\}$ mutually independent [1,2,3]. The other idea is to maximize the entropy (Informax) $J(W) = -\int_z p(z) \ln p(z) dz$ with $z = [p_1(y_1), \cdots, p_n(y_n)]$ [4]. Both are shown to be special cases of a general ICA framework and are equivalent when the accurate $p_i(y_i)$ is available [9]. However, the accurate $p_i(y_i)$ is difficult to get. In [2,3], it is approximated by a pre-fixed truncated Edgeworth series or truncated Gram-Charlier series. In [4], it is simply fixed at pre-given densities $p_i(y_i) = \frac{ds(y_i)}{dy_i}$ with $s(y_i)$ being one of those sigmoid functions used in the literature of neural networks, and understandably such an idea will only work for some special type of source signal (e.g., super-gaussians). In [10], a new strategy is suggested, in which $p_i(y_i)$ is no longer prefixed but learned via a flexible density function based a finite mixture of densities. It has been shown experimentally that this new method can work well in not only the examples that the methods in [1,3,4] work, but also the examples that the methods in [1,3,4] fail.

*Supported by This project was supported by the HK RGC Earmarked Grants CUHK250/94E, CUHK484/95E, and Ho Sin-Hang Education Endowment Fund HSH 95/02. Email lxu@cs.cuhk.hk

0-7803-4256-9/97/$10.00 ©1997 IEEE

The present author has a strong belief that the direction of trying approximates $p_i(y_i)$, though justified, is over-necessary. Therefore, in [9] the marginal density $p_i(y_i)$ was actually replaced by a general integrable function $g_i(y_i)$, while what kind of this function should be remained unclear. In [11], some examples of $g_i(y_i)$ that is able to implement ICA under certain conditions have been further given with theoretical analyses. Of course, not every of $g_i(y_i)$ can success in ICA, the workable $g_i(y_i)$ must come from a family $\mathcal{G}$. However, what kinds of properties this family should bear still remain as open a question.

In recent years, a so called *Bayesian Ying-Yang (BYY) Learning* system and theory has been developed as a unified statistical learning approach, which can provide us at least four types of new strengths [5,6,7,8]. **First**, it is able to unify most of the existing major statistical learning models and theories. For examples, for unsupervised learning they include ML learning with the EM algorithm, information geometry theory with the *em* algorithm, MDL autoencoder, Helmholtz machine, independent component analysis (ICA) by INFORMAX or MMI, LMSER learning, principal component analysis (PCA), various clusters and self-organizing maps; and for supervised learning they include the conventional ML learning (i.e. BP algorithm) for feed-forward network, ML learning for RBF nets, mixture of experts and its alternatives. This powerful unification provides us not only deep insights on these mentioned popular existing approaches but also further guidances on obtaining their new variants or extensions via cross-fertilization. **Second**, some special cases of the *BYY Learning* bring us several interesting new models on both unsupervised and supervised learnings, which deserve further investigation. **Third**, the *BYY Learning* theory can function as a general theory not only for parameter learning , but also for model selection (or more precisely called structural scale selection), e.g., for selecting subspace dimension, number of clusters, number of gaussians, number of experts, number of hidden units, etc. **Finally**, this same theory can also function as a general theory from regularization and architecture evaluation. Readers are referred to [5,6] for a rather systematic review on previous results and several new advances.

In this paper, we show that a particular case of the BYY learning system and theory will reduce into a very general enhanced information-theoretic ICA framework with several new powers. After briefly introducing the BYY learning system and theory in Sec.2, we propose this ICA frame work in Sec.3. Then, in Sec.4 the gradient-based ICA algorithms are developed in its general forms and two detailed implementations. In Sec.5, the algorithms for two important special cases are investigated in details with three important theorems. Finally, in Sec.6 an variant ICA algorithm is proposed and demonstrated to be more robust for outliers.

2. BYY Learning System and Theory

The perception tasks can be summarized into the problem of estimating joint distribution $p(x, y)$ of the observable pattern x in the observable space X and its representation pattern y in the representation space Y, as shown in Fig.1. We call a passage $M_{y|x}$ for the flow like $x \rightarrow y$ a *Yang*/(male) passage since it performs the task of transferring a pattern/(a real body) into a code/(a seed). We call a passage $M_{x|y}$ for the flow $y \rightarrow x$ as a *Ying*/(female) passage since it performs the task of generating a pattern/(a real body) from a code/(a seed). $M_{y|x}$ and $M_{x|y}$ are complement to each other and together implement an entire circle $x \rightarrow y \rightarrow x$. Interestingly, under the Bayesian framework we also have two representations $p(x, y) = p(y|x)p(x)$ and $p(x, y) = p(x|y)p(y)$. We use a Yang/(visible) model M_x representing $p(x)$ (i.e., modeling the space X), and we use a Ying/(invisible) model M_y representing $p(y)$ (i.e., modeling the space Y). Moreover, $M_{y|x}$ is represented

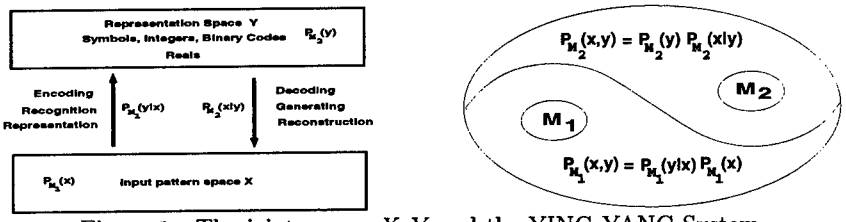


Figure 1 The joint spaces X, Y and the YING-YANG System

by $p_{M_{y|x}}(y|x)$ and $M_{x|y}$ by $p_{M_{x|y}}(x|y)$. Together, we have a *YANG* machine $M_1 = \{M_{y|x}, M_x\}$ to implement $p_{M_1}(x,y) = p_{M_{y|x}}(y|x)p_{M_x}(x)$ and a *YING* machine $M_2 = \{M_{x|y}, M_y\}$ to implement $p_{M_2}(x,y) = p_{M_{x|y}}(x|y)p_{M_y}(y)$. A pair of YING-YANG machines is called a YING-YANG pair or a YING-YANG system. Such a formalization compliments to a famous Chinese ancient philosophy that *every entity in universe involves the interaction between YING and YANG.*

The task of specification of a Ying-Yang system is called *learning* in a broad sense. First, we need to specify the variables x, y. Usually, x is assumed to be $x \in R^d$. But y can be an $y \in R^{k_r}$, an integer $y \in [1, 2, \cdots, k_r]$ and a binary $y = [y_1, \cdots, y_{k_r}]$, $y_i \in [0, 1]$, where k_r represents the scale or complexity of representation. Next, we specify four components $p_{M_x}(x)$, $p_{M_{y|x}}(y|x)$, $p_{M_{x|y}}(x|y)$ and $p_{M_y}(y)$. Generally speaking, each of them is specified by three parts. The **first** part is called *Architecture Design*, denoted by S_a, consisting of the general setting on (a) its density function form p (.), (b) one or several types of basic structural units and (c) an architecture for organizing a number of these structural units. The **second** part is called model selection or more precisely *Structural Scale Selection* for selecting scale parameters $k = [k_r, k_a^b]$ which consists of the above representation scale k_r and the scale or complexity k^b of those more complicated basic structural units themselves. The **third** part is called *Parameter Learning or Estimation*, also often called *learning* simply in a narrow sense, for specifying a particular value of θ — a set of real variables on certain domain. Together, for each $a \in \{x, x|y, y|x, y\}$, each $M_a = \{S_a, \theta_a, k\}$ is specified only after all its three parts are specified. Some examples are given in [7] to this formalization better.

Our basic theory is that the specifications of the three levels, namely *Architecture Design, Structural Scale Selection* and *Parameter Learning* should best enhance the so called *Ying-Yang Harmony or Marry*, through minimizing a so called *separation functional*:

$$F_s(M_1, M_2) = F_s(p_{M_{y|x}}(y|x)p_{M_x}(x), p_{M_{x|y}}(x|y)p_{M_y}(y)) \geq 0,$$
$$F_s(M_1, M_2) = 0, \ \textit{if and only if} \ p_{M_{y|x}}(y|x)p_{M_x}(x) = p_{M_{x|y}}(x|y)p_{M_y}(y), \qquad (2)$$

which describes the harmonic degree of the Ying-Yang pair. Such a learning system and theory is called as *Bayesian Ying-Yang (BYY) Learning System and Theory.*

Three categories of separation functionals, namely *Convex Divergence, L_p Divergence, and De-correlation Index*, have been suggested in [6]. The *Convex Divergence* is defined as

$$F_s(p_1, p_2) = f(1) - \int_x p_1(x) f(\frac{p_2(x)}{p_1(x)})dx \geq 0, \quad f(u) \ \textit{is a strict convex on} \ (0, +\infty), \qquad (3)$$

from which we get $F_s(M_1, M_2)$ by substituting p_1 with $p_{M_{y|x}}(y|x)p_{M_x}(x)$ and p_2 with $p_{M_{x|y}}(x|y)p_{M_y}(y)$. Its three typical examples are given as follows:

(a) $f(u) = \ln u$, which leads us to the Kullback Divergence:

$$KL(M_1, M_2) = \int_{x,y} p_{M_{y|x}}(y|x)p_{M_x}(x) \ln \frac{p_{M_{y|x}}(y|x)p_{M_x}(x)}{p_{M_{x|y}}(x|y)p_{M_y}(y)} dxdy. \qquad (4)$$

In the special case, the BYY learning is called Bayesian-Kullback YING-YANG (BKYY) learning. It is a most useful case and has been extensively studied [5,6,7,8].

(b) $f(u) = -u^\beta, \beta > 1$, called as Minus Convex divergence.

(c) When $f(u) = u^\beta, 0 < \beta < 1$, we called as Positive Convex (PC) divergence. Interestingly, when $\beta = 0.5$, it leads to a Root-Inner-Product (RIP) divergence $F_s(p_1, p_2) = 1 - \int_x \sqrt{p_1(x)p_2(x)}dx$, which has a nice symmetric feature that the Kullback divergence does not have.

From the BKYY learning eq.(4), we can obtained those four new strengths mentioned in Sec.1 for unsupervised learning. Moreover, by replacing the Kullback divergence eq.(4) with the above mentioned other non-Kullback separation functionals, we can obtain various alternatives for those Kullback divergence based learning models, with some new interesting properties (e.g., robust learning), we call them a Bayesian Non-Kullback separation functionals Ying-Yang (BNYKK) learning. Also, the BYY learning eq.(5) can be extended into a more general BYY learning system such that those four new strengths mentioned in Sec.1 for supervised learning can also obtained [12,13], both for the cases that the Kullback divergence eq.(1) is used and for the cases that those non-Kullback separation functionals are used.

In the rest of this paper, we will only concentrate on a particular case of the BYY learning system and theory that leads to ICA problem.

3. General BYY ICA Framework and BKYY ICA Scheme

We modify the problem eq.(1) into a more general case that sensors are affected by noises and the number m of sensors may be larger than the number n of sources. That is,

$$x = As + e_x, s = y + e_y,\ y = Wx,\ m \geq n, \tag{5}$$

where the dimension for y, s vectors is n and for x is m, also we have that $Es = 0$, the m components of e_x are mutually independent with $Ee_x = 0$ and also independent of s, and thus also $Ex = 0$, $Ey = 0$ and $Ee_y = 0$.

Here we consider that the Ying space is for s and the Yang space is for x. The Yang density is $p_{M_x}(x) = p(x)$ which can be estimated from the data by sensors. The Ying density is unknown exactly, but we can assume that $p_{M_s}(s) = g(s, \theta) = \prod_{i=1}^n g_i(s_i, \theta_i)$ with $g_i(s_i, \theta_i)$ being of simple density function form specified but θ_i is variable in Θ that actually represents the family $\mathcal{G}$. The Yang passage $s = y + e_y,\ y = Wx$ is described by a density $p_{M_{s|x}}(s|x)$ for the random variable e_y under each given x (thus y). The Ying passage $x = As + e_x$ is described by a density $p_{M_{x|s}}(x|s)$ for the random variable e_x under each given x. Furthermore, we can assume that e_x is from gaussian $G(e_x, 0, \sigma^2 I)$, and thus $p_{M_{x|s}}(x|s) = G(x, As, \sigma^2 I)$. Moreover, the fact $s = WAs + We_x + e_y$ suggests that a reasonable case is

$$WA = I, \quad and\ thus\ A = W^- = W^T(WW^T)^{-1},\ e_y = -We_x. \tag{6}$$

Thus, $p_{M_{x|s}}(x|s) = G(x, W^- s, \sigma^2 I)$, and e_y is a gaussian $G(e_y, 0, WW^T\sigma^2 I)$ and independent of x.

In summary, we have specified the architectural design as follows

$$p_{M_x}(x) = p(x),\ p_{M_{s|x}}(s|x) = G(s, Wx, \sigma^2 WW^T),$$
$$p_{M_{x|s}}(x|s) = G(x, W^- s, \sigma^2 I), p_{M_s}(s) = g(s, \theta) = \prod_{i=1}^n g_i(s_i, \theta_i). \tag{7a}$$

Next, according to the Ying-Yang learning theory, we specify the remaining un-specified items, namely, (a) the structural scale $k = k_r = n$, i.e., the number of sources; (b) σ^2; (c) W; and (d) θ, via eq.(2) to minimize $F_s(M_1, M_2)$:

$$\{W^*, \sigma^{*2}, n^*, \theta^*\} = arg\min_{\{W, \sigma^2, n, \theta \in \Theta\}} J(W, \sigma^2, n, \theta),$$
$$J(W, \sigma^2, n, \theta) = \int_{x,s} F_s(G(s, Wx, \sigma^2 WW^T)p(x), G(x, W^- s, \sigma^2 I)g(s, \theta))dxds. \tag{7b}$$

which is called BYY ICA framework. Particularly, when the Kullback divergence eq.(4) is used, it is simplified into

$$\{W^*, \sigma^{*2}, n^*, \theta^*\} = arg\min_{\{W, \sigma^2, n, \theta \in \Theta\}} J(W, \sigma^2, n, \theta), \quad J(W, \sigma^2, n, \theta) =$$
$$\int_{x,s} G(s, Wx, \sigma^2 WW^T)p(x) \ln \frac{G(s, Wx, \sigma^2 WW^T)p(x)}{G(x, W^- s, \sigma^2 I)g(s, \theta)} dxds. \tag{8}$$

Which is called BKYY ICA Scheme. Due to the space limit, the theoretical justification of eq.(7b) and eq.(8) is given elsewhere [14]. Theoretical justification for special cases will be given in Sec.5. From eq.(8), we see that

$\{W, \sigma^2, n, \theta\}$ should be irrelevant to the term $\int_x p(x) \ln p(x) dx$ and thus can be omitted. As shown in details in [14], we can equivalently re-write eq.(8) into

$$J_s(W, \sigma^2, n, \theta) = 0.5[(m-n) \ln \sigma^2 - \ln |WW^T| + m - n] + EJ_G(x, W, \sigma^2, \theta),$$
$$J_G(x, W, \sigma^2, \theta) = -\int_s G(s, Wx, \sigma^2 WW^T) \ln g(s, \theta) ds, \quad g(s, \theta) = \prod_{i=1}^n g_i(s_i, \theta_i),$$
$$\{W_n^*, \theta_n^*\} = arg \min_{\{W, \theta \in \Theta\}} J_s(W, \sigma^2, n, \theta), \quad s.t. \quad \sigma^2 = \frac{1}{m-n} E\|x - W^- Wx\|^2$$
$$\sigma_n^{*2} = \frac{1}{m-n} E\|x - W_n^{*-} W_n^* x\|^2, \quad n^* = arg \min_n \ J(n), \quad J(n) = J_s(W_n^*, \sigma_n^{*2}, n, \theta_n^*). \quad (9)$$

With this BKYY ICA scheme, the best parameters $W_n^*, \theta_n^*, \sigma_n^{*2}$ are estimated for each given n, and then a best number n^* of sources is found via $n^* = arg \min_n J(n)$ with the corresponding $W_{n^*}^*, \theta_{n^*}^*, \sigma_{n^*}^{*2}$ as the final best estimations.

4. The Gradient-Based Algorithms for The BKYY ICA

To practically implement BKYY ICA, the key is how to to get $W_n^*, \theta_n^*, \sigma_n^{*2}$ for each given n. Here, we adopt the gradient-based method. For simplicity, we will omit the subscript n in any cases without confusions.

First, we consider $J_G(x, W, \sigma^2, \theta)$ by Taylor expansion of $\ln g(s, \theta)$ around $y = Wx$, and then use the expansion to do the integral:

$$J_G(x, W, \sigma^2, \theta) = -\int_s G(s, Wx, \sigma^2 WW^T) \ln g(s) ds$$
$$= -[\ln g(Wx, \theta) + \tfrac{1}{2}\sigma^2 tr(D_h WW^T) + o(\sigma^2)],$$
$$h(s, \theta) = \frac{\partial g(s,\theta)/\partial s}{g(s,\theta)}, \quad D_h = \partial h(s, \theta))/\partial \theta^T. \quad (10a)$$

For simplicity, we approximately just use the first term $-\ln g(Wx, \theta)$ which is valid when σ^2 is small. Therefore, we have

$$\partial J_G(x, W, \sigma^2(W), \theta)/\partial W = -h(Wx, \theta) x^T. \quad (10b)$$

Next, we have that $\sigma^2 = \frac{1}{m-n} E\|x - W^- Wx\|^2 =$

$$= tr[R_x] - tr[R_x W^T (WW^T)^{-1} W], \quad R_x = E(xx^T),$$
$$\frac{d\sigma^2}{dW} = \frac{1}{m-n} 2(WW^T)^{-1}[WR_x - WR_x W^T (WW^T)^{-1} W].$$

Together with $\frac{d \ln |WW^T|}{dW} = 2(WW^T)^{-1} W$, it follows from eq.(9) that

$$\frac{dJ_s(W, \sigma^2, n, \theta)}{dW} = (WW^T)^{-1}[\frac{WR_x - WR_x W^T (WW^T)^{-1} W}{\sigma^2} - W] - E[h(Wx, \theta) x^T].$$

Therefore, we can get the following general forms of both the batch way and adaptive gradient algorithms for updating W, σ^2 with a stepsize η:

$$\Delta W = -\eta\{\Delta_{W_\sigma^2} - (WW^T)^{-1} W - E[h(Wx, \theta) x^T]\},$$
$$\Delta_{W_\sigma^2} = (WW^T)^{-1} \frac{WR_x - WR_x W^T (WW^T)^{-1} W}{\sigma^2},$$
$$\sigma^2 = \frac{1}{m-n} E\|x - W^T (WW^T)^{-1} Wx\|^2. \quad (11)$$

$$\Delta W = -\eta\{\Delta_{W_\sigma^2} - (WW^T)^{-1} W - h(Wx, \theta) x^T\},$$
$$\sigma^{new\ 2} = \sigma^{old\ 2} + \eta \frac{1}{m-n} \|x - W^T (WW^T)^{-1} Wx\|^2. \quad (12a)$$

Similarly, we can get the general forms of both batch way and adaptive algorithms for updating θ by

$$\Delta\theta = \eta E \frac{\partial \ln g(Wx,\theta)}{\partial \theta}, \quad \Delta\theta = \eta \frac{\partial \ln g(Wx,\theta)}{\partial \theta}. \quad (12b)$$

Thus, our gradient algorithm can be summarized as an iterative process that in each iteration both eq.(11) and eq.(12b) are used for updating W, σ^2 and θ. As well known, as long as the learning stepsize controlled appropriately, a gradient descent iterative process will guarantee to converge to at least a local minimum of $J_s(W, \sigma^2, n, \theta)$. So, our iterative process is guaranteed to converge, and then we use the converged results as our estimates $W_n^*, \theta_n^*, \sigma_n^{*2}$.

To get further detailed algorithms, we focus on two types of $g(s, \theta)$ family. The first one is the finite mixture of parametric densities

$$g(s, \theta) = \sum_{j=1}^q \alpha_j p(s|\lambda_j), \quad \sum_{j=1}^q \alpha_i = 1, \alpha_i \geq 0,$$
$$\theta = [\lambda_1 \cdots, \lambda_q]^T, \quad p(s|\lambda_j) = \prod_{i=1}^n p(s_i|\lambda_{j,i}), \quad (13a)$$

where $p(s_i|\lambda_{j,i})$ is some simple density function. In this case, we have

$$h(s,\theta)=\sum_{j=1}^{q}p(j|s)h(s,\lambda_j),\quad h(s,\lambda_j)=\frac{\partial p(s|\lambda_j)}{\partial s}/p(s|\lambda_j),$$
$$p(j|s)=\frac{\alpha_j p(s|\lambda_j)}{\sum_{j=1}^{q}\alpha_j p(s|\lambda_j)},\quad \alpha_j=\int_s p(j|s)ds. \tag{13b}$$

For a gaussian $p(s|\lambda_j)=G(s,0,\Lambda_j)$ with Λ_j being a diagonal matrix , we have

$$h(s,\lambda_j)=-\Lambda_j^{-1}s_i,\quad \frac{\partial \ln g(Wx,\theta)}{\partial \Lambda_j}=-\frac{1}{2}p(j|s)[\Lambda_j^{-1}-\Lambda_j^{-1}Wxx^TW^T\Lambda_j^{-1}]. \tag{14}$$

Thus, putting them into eq.(12), we get the batch way gradient algorithm:

$$\Delta W=-\eta\{\Delta_{W_\sigma^2}-(WW^T)^{-1}W+\sum_{j=1}^{q}\Lambda_j^{-1}WE(p(j|Wx)xx^T)\}; \tag{15a}$$
$$p(j|Wx)=\frac{\alpha_j p(Wx|\lambda_j)}{\sum_{j=1}^{q}\alpha_j p(Wx|\lambda_j)}, \tag{15b}$$
$$\alpha_j=\int_x p_h(x)p(j|Wx)dx,\quad \Lambda_j=WE[p(j|Wx)xx^T]W^T. \tag{15c}$$

Actually, the algorithm for θ by eqs.(15b&c) is the so called EM algorithm. Also, with eq.(15b) still the same, we have the adaptive algorithm given by

$$\Delta W=-\eta\{\Delta_{W_\sigma^2}-(WW^T)^{-1}W+\sum_{j=1}^{p}\Lambda_j^{-1}p(j|Wx)xx^T\}, \tag{16a}$$
$$\alpha_j^{new}=\alpha_j^{old}+\eta p(j|Wx),\quad \Lambda_j^{new}=\Lambda_j^{old}+\eta p(j|Wx)Wxx^TW^T. \tag{16b}$$

The second $\mathcal{G}$ family is called *4-th order exponential-polynomial family*:

$$p(s)=\sum_{i=1}^{n}p_i(s)=-\frac{1}{2}s^T\Lambda_2^{-1}s+\frac{1}{3}s^T\Lambda_3^{-1}s^2-\frac{1}{4}s^T\Lambda_4^{-1}s^3$$
$$g(s,\theta)=c_0^{-1}exp(p(s)),\quad s^k=[s_1^k,\cdots,s_n^k]^T,k=2,3, \tag{17}$$

where $\Lambda_k,k=2,3,4$ are diagonal matrices, which are prefixed such that the following condition holds:

$$c_0=\int_s exp(p(s))ds<\infty,\quad \int_s sg(s,\theta)ds=0.$$

Thus, from eq.(12), we can get the algorithm for updating W:

$$\Delta W=-\eta\{\Delta_{W_\sigma^2}-(WW^T)^{-1}W+E[\{\Lambda_2^{-1}Wx-\Lambda_3^{-1}(Wx)^2+\Lambda_4^{-1}(Wx)^3\}x^T]\}. \tag{18}$$

Also, from eq.(10a) we have that for $k=2,3,4$

$$\frac{\partial \ln g(s,\theta)}{\partial \Lambda_k}=-c_0^{-1}\frac{\partial c_0}{\partial \Lambda_2}+\frac{\partial p(s)}{\partial \Lambda_k}=\frac{1}{2}\Lambda_k^{-1}[-\int_s g(s,\theta)s^ks^Tds+s^ks^T]\Lambda_k^{-1}.$$

Unfortunately, the gradient is difficult to be used since $\int_s g(s,\theta)s^ks^Tds$ is difficult in computation, although it is well defined. So, $\Lambda_k,k=2,3,4$ is prefixed above.

5. BKYY ICA in Two Special Cases

The first special case is that there is no noise, i.e., $x=A^os^o$ with $e_x^o=0$. The algorithms given in Sec.4 still works here with an interesting property.

When $m\neq n$, from the fact $(m-n)\sigma^2=E\|x-W^-Wx\|^2tr[E(s^os^{oT})A_o^T(I-W^-W)^T(I-W^-W)A_o]$, and the fact that $E(s^os^{oT})$ is diagonal, we have that $\sigma^2=0$ if and only if $tr[A_o^T(I-W^-W)^T(I-W^-W)A_o]=0$ or equivalently $(I-W^-W)A_o=(I-W^T(WW^T)^{-1}W)A_o=0$, which is true if and only if the space spanned by A_o is contained in the space spanned by W. In other words, $A_o=W^TB$ with $n\geq n_o$ and B is an $n\times n_o$ matrix. In this case from eq.(9), we know $\sigma^2=0$, $WA_oE(s^os^{oT})B^TW-WA_oE(s^os^{oT})B^TW=0$. Thus, in eq.(9) we have $J_G(x,W,\sigma^2,\theta)\to-\infty$, and in eq.(11) and also in the subsequent related equations we have $WR_x-WR_xW^T(WW^T)^{-1}W\sigma^2$ undefined. Apparently, our method can not normally work here.

In fact, this is just an interesting property for fast detecting the number n_o of sources in the situation without noise affects. In other words, for each n we can run any of our algorithms in Sec.4 until either normal convergence or $\sigma^2=0$, and then we can get the correct n_o by simply increasing from the lower side to n_o until σ^2 becomes zero (or very small) or from the upside to n_o until σ^2 is no longer regarded as zero.

After we get this n_o, we can simply drop the term $(m-n)\ln\sigma^2$ in eq.(9). Moreover, since $\sigma^2=0$, $J_s(W,\sigma^2,n,\theta)=-\ln g(Wx,\theta)$ by eq.(10a) is exact without any approximation anymore. Thus, eq.(9) becomes

$$J_s(W,\theta)=-0.5\ln|WW^T|-E\ln g(Wx,\theta),\quad \{W^*,\theta^*\}=arg\min_{\{W,\theta\in\Theta\}}J_s(W,\theta). \tag{19}$$

and eq.(12a) becomes the following eq.(20) with eq.(12b) used still for θ:

$$\Delta W=\eta\{(WW^T)^{-1}W+E[h(Wx,\theta)x^T]\},\quad \Delta W=\eta[(WW^T)^{-1}W+h(Wx,\theta)x^T] \tag{20}$$

Also, for the $\mathcal{G}$ given by eq.(13a), we have that eq.(15a) and eq.(16a) become the following eq.(21) with eq.(15c) and eq.(16b) used still for θ, respectively:

$$\Delta W = \eta\{(WW^T)^{-1}W - \sum_{j=1}^{q} \Lambda_j^{-1} W E[p(j|Wx)xx^T]\};$$
$$\Delta W = \eta[(WW^T)^{-1}W - \sum_{j=1}^{q} \Lambda_j^{-1} W p(j|Wx)xx^T]. \quad (21)$$

Furthermore, for the $\mathcal{G}$ given by eq.(17), it follows that eq.(18) becomes:

$$\Delta W = \eta\{(WW^T)^{-1}W - E[\{\Lambda_2^{-1}Wx - \Lambda_3^{-1}(Wx)^2 + \Lambda_4^{-1}(Wx)^3\}x^T]\}. \quad (22)$$

In the above first special case, if we further let $m = n = n_o$, we get the second special case. In fact, it is the case given by eq.(1) that has been widely assumed in the literature. In this case, W, A_o are both $n_o \times n_o$ nonsingular matrices. Thus, we have $|WW^T| = |W|^2$ and eq.(19) further becomes

$$J_s(W,\theta) = -\ln|W| - E\ln g(Wx,\theta), \quad \{W^*,\theta^*\} = arg\min_{\{W,\theta\in\Theta\}} J_s(W,\theta). \quad (23)$$

which is exactly the so called information-theoretic approach or maximum likelihood method [1,2,3,4]. Moreover, we have that eq.(20) becomes

$$\Delta W = \eta[(W^{-1})^T + E(h(Wx,\theta)x^T)], \quad \Delta W = \eta[(W^{-1})^T + h(Wx,\theta)x^T] \quad (24a)$$

When $h(Wx,\theta)$ is pre-fixed, it is exactly the INFORMAX algorithm by [4]. Via modifying ΔW by $W\Delta WW^T$, eq.(24a) becomes the natural gradient algorithm for MMI[3]:

$$\Delta W = \eta[W + WE(h(Wx,\theta)(Wx)^T)], \quad \Delta W = \eta[W + Wh(Wx,\theta)(Wx)^T] \quad (24b)$$

In [3], $g(Wx,\theta)$ (thus $h(Wx,\theta)$) is pre-fixed via a truncated Gram-Charlier series. Furthermore, by using eq.(24a) together with eq.(12b) for learning $g(Wx,\theta)$ via updating θ, with $g(Wx,\theta)$ defined by eq.(13a) and $p(s_i|\lambda_{j,i})$ being the derivative of a simple sigmoid function, then we get the so called *Learned Mixture of Parametric Densities* for the information-theoretic approach [10,11].

In the rest of this section, we further propose two algorithms. The first one is obtained via modifying ΔW by $W\Delta WW^T$ in eq.(21) and together using eq.(15c) for updating θ. That is, we have

$$\Delta W = \eta\{W - W\sum_{j=1}^{q} \Lambda_j^{-1} E[p(j|Wx)(Wx)(Wx)^T]\}; \quad (25a)$$

$$p(j|Wx) = \frac{\alpha_j p(Wx|\lambda_j)}{\sum_{j=1}^{q} \alpha_j p(Wx|\lambda_j)}, \quad \alpha_j = Ep(j|Wx), \quad \Lambda_j = WE[p(j|Wx)xx^T]W^T. \quad (25b)$$

$$\Delta W = \eta[W - W\sum_{j=1}^{q} \Lambda_j^{-1} p(j|Wx)(Wx)(Wx)^T]. \quad (25c)$$

$$\alpha_j^{new} = \alpha_j^{old} + \eta p(j|Wx), \quad \Lambda_j^{new} = \Lambda_j^{old} + \eta p(j|Wx)Wxx^TW^T, \quad (25d)$$

which is a new *Learned Gaussian Mixture* algorithm for the problem eq.(1). Its advantages are shown in the following two theorems.

Theorem 1 *For the problem eq.(1) and using the batch algorithm eqs.(25a&b) with $q = 2$ only, given the converged nonsingular W^* and other parameters $p^*(j|W^*x), \Lambda_j^*$ as well as $V^* = W^*A_o$, denote $R_j^x = E[p^*(j|W^*x)xx^T]$ and $Rank[R_1^x - R_2^x] = k_r$. Then, we have that $V^* = \Pi D$ as long as $k_r \geq n_o - 1$, where Π is a permutation matrix and D being a non-singular diagonal matrix. Moreover, when $k_r < n_o - 1$, there are k_r row vectors of V^* that are linear independent and each of them contains only one nonzero elements.*

Proof From eq.(25b), after converged we have $W^*R_j^xW^{*T} = \Lambda_j, j = 1,2$ with Λ_j being a positive diagonal matrix. From $R_j^x = A_oR_j^sA_o^T$ with A_o being full rank, we know $Rank[R_1^x - R_2^x] = Rank[R_1^s - R_2^s] = k_r$. Also, since that $R_j^s = E[p^*(j|W^*A_o)ss^T]$ should be a positive diagonal matrix too, we know that $R_1^s - R_2^s$ has k_r nonzero diagonal elements, which means that there are k_r corresponding different diagonal elements between R_1^s and R_2^s, that is, $R_2^s(R_1^s)^{-1}$ has k_r different diagonal values. Furthermore from $\Lambda_j = W^*A_oR_j^s(W^*A_o)^T = V^*R_j^sV^{*T}$, we have $R_1^sV^{*T} = V^{*-1}\Lambda_1$, and $V^{*-1}\Lambda_2 = R_2^sV^{*T} = R_2^s(R_1^s)^{-1}R_1^sV^{*T} = R_2^s(R_1^s)^{-1}V^{*-1}\Lambda_1$. Let's denote $R = R_2^s(R_1^s)^{-1}$ and $\Lambda = \Lambda_2\Lambda_1^{-1}$, which are both diagonal, we have $RV^{*T} = V^{*T}\Lambda$. For the j-th column vector of V^{*T} or the j-th row vector of V^*, we have $Rv_j^T = \lambda_j v_j^T$ with λ_j being the j-th digonal element of Λ . For those k_r different diagonal elements of R, we have that the k_r corresponding vectors v_j^T's are linear independent with each containing only one nonzero element. When $n_o - k_r = 1$, the remaining v_j^T should also contain only one nonzero element. Therefore, we have $V^* = \Pi D$ as long as the rank $k_r \geq n_o - 1$. When $k_r < n_o - 1$, since there are

$n_o - k_r$ diagonal elements of R is 1, their corresponding $n_o - k_r$ vectors for v_j^T can be linear independent but each contains more than one nonzero elements. **Q.E.D.**

According to this theorem, when the source s is gaussian, we have the learned $R_1^x = R_2^x$ and $k_r = 0$, the above Learned Gaussian Mixture algorithm will fail. When there are $n_o - k_r$ sources in s are gaussian, then $Rank[R_1^x - R_2^x] = k_r$ and the algorithm can not recover the $n_o - k_r$ gaussian sources and the other k_r sources can be recovered. If there is at most one gaussian source in s, it usually will lead to $Rank[R_1^x - R_2^x] = k_r \geq n_o - 1$ and we have $V^* = \Pi D$. This theorem has justified the experimental successes given in [10,11].

This theorem has also provided an implementable way for checking whether the obtained result is fully successful or partially, or fail through checking $Rank[R_1^x - R_2^x] = k_r$. In practice, numerical error or other fact may affect the accurate of the algorithm and the estimate of $Rank[R_1^x - R_2^x] = k_r$. For reliability, we can use a gaussian mixture with $q > 2$. In this case, we can get the following Theorem 2 based on the above theorem.

Theorem 2 *For the problem eq.(1) and using the batch algorithm eqs.(25a&b) with $q > 2$, given the converged nonsingular W^* and other parameters $p^*(j|W^*x), \Lambda_j^*$ as well as $V^* = W^*A_o$, denote $R_j^x = E[p^*(j|W^*x)xx^T]$ and k_r is the rank of S with $S = \cup_{\{all\ i,j\}} S_{i,j}$ and $S_{i,j}$ being the subspace spanned by the column vectors of $R_i^x - R_j^x$. Then, we have that $V^* = \Pi D$ as long as $k_r \geq n_o - 1$, where Π is a permutation matrix and D being a non-singular diagonal matrix. Moreover, when $k_r < n_o - 1$, there are k_r row vectors of V^* that are linear independent and each of them contains only one nonzero elements.*

Theorem 2 suggests that as long as q is large enough, we can fully success if there is at most one gaussian source in s; otherwise, we can still recover those non-gaussian sources.

Next, we give the second algorithm for the problem eq.(1), which is obtained by modifying ΔW by $W\Delta WW^T$ in eq(18), we have

$$\Delta W = \eta\{W - WE[\{\Lambda_2^{-1}Wx - \Lambda_3^{-1}(Wx)^2 + \Lambda_4^{-1}(Wx)^3\}(Wx)^T]\}, \tag{26}$$

for which we have

Theorem 3 *For the problem eq.(1) and assuming that the batch algorithm eqs.(26) converged with nonsingular W^* and $V^* = W^*A_o$. Denote $R_s^j = diag[E(s_1^j), \cdots, E(s_n^j)], j = 2,3,4$. Then, we have $V^* = \Pi D$ as long as R_s^3 is full rank and $off_{diag}\{\Lambda_k^{-1}E[(W^*x)^k(W^*x)^T]\} \neq off_{diag}\{\Lambda_j E[(W^*x)^j(W^*x)^T]\}$ unless $off_{diag}\{E[(W^*x)^k(W^*x)^T]\} = off_{diag}\{E[(W^*x)^j(W^*x)^T]\} = 0$, where $j \neq k \in [2,3,4]$ and $off_{diag}[A]$ denotes all the off-diagonal elements of A.*

Proof From eq.(26), after converged we have $I = \Lambda_2^{-1}W^*Exx^TW^{*T} - \Lambda_3^{-1}E[(W^*x)^2(W^*x)^T] + \Lambda_4^{-1}E[(W^*x)^3)(Wx)^T]$, and under the condition of the theorem, we have $E[(W^*x)^k(W^*x)^T] = D_k$, for $k = 2,3,4$ with D_k being diagonal. First, from $E[(W^*A_o)ss^T(W^*A_o)^T] = V^*R_s^2V^{*T} = D_2$ with R_s^2 being a positive definite diagonal matrix, and we have $V^* = D_2^{1/2}\Phi^T(R_s^2)^{-1/2}$ with $\Phi^T\Phi = I$. Second, from $E[(W^*x)^2(W^*x)^T] = E[(V^*s)^2(V^*x)^T] = D_3$, we have $V_{(2)}^*R_s^3V^{*T} = D$ with D being diagonal and $V_{(2)}^* = [v_{i,j}^2]$ (i.e., its each element is the square of $v_{i,j}$ of $V = [v_{i,j}]$). Moreover, with $V^* = D_2^{1/2}\Phi^T(R_s^2)^{-1/2}$, we have $V_{(2)}^* = [v_{i,j}^2] = D_2^2[\phi_{i,j}^2](R_s^2)^{-1}$ with $\Phi^T = [\phi_{i,j}]$ and $\Phi^T\Phi = I$ as given above. Therefore $R_s^2[\phi_{i,j}^2](R_s^2)^{-1}R_s^3(R_s^2)^{-1/2}\Phi(R_s^2)^{1/2} = D$, or $[\phi_{i,j}^2]R\Phi = D'$ with R and D' being both diagonal. It further follows that $\Phi = R^{-1}[\phi_{i,j}^2]^{-1}D'$ and $I = \Phi^T\Phi = D'[\phi_{i,j}^2]^{-T}R^{-2}[\phi_{i,j}^2]^{-1}D'$ or by inverting it becomes $D'^2 = [\phi_{i,j}^2]R^2[\phi_{i,j}^2]^T$, which means $[\phi_{i,j}^2] = D'\Psi R^{-1}$ with $\Psi^T\Psi = I$. Thus, $D' = [\phi_{i,j}^2]R\Phi = D'\Psi R^{-1}R\Phi$ or $\Psi\Phi = I$. That is, $\Psi = \Phi^T$ and $[\phi_{i,j}^2]R = D'\Phi^T = D'[\phi_{i,j}]$, which means that $r_i\phi_{i,j}^2 = \phi_{i,j}d_j'$, thus either $\phi_{i,j} = 0$ or $\phi_{i,j} = d_j'/r_i$, with r_i and d_j' being the diagonal element of R, D' respectively. Therefore, $\Phi^T = RED'$ and each element of E can only to be 0 or 1. It follows from $I = \Phi^T\Phi = RED'^2E^TR$ that $R^{-2} = ED'^2E^T = \sum_{i=1}^{n_o} d_i'^2e_ie_i^T$ with e_i is the column of E. Thus, the off-diagonal elements of $e_ie_i^T$ must be zeros, or equivalently E is a permutation matrix $\Phi^T = EERED' = ED_r$ with D_rERED' being still diagonal. **QED.**

The above theorem suggests a way to improve the algorithm eq.(26) via imposing some

constraints to enhance the chance of satisfying the condition of the theorem. Due to the limited space, we leave the details elsewhere.

Not only the above algorithms given by eq.(25) and eq.(26) have extended the existing information-theoretic approaches [1,2,3,4], but also the algorithms given by eq.(20), eq.(21) and eq.(22) can be also regarded as the extended information-theoretic approaches for the cases that the number of sensors are larger than the unknown number of sources. Finally, all these approaches are the special cases of the BKYY ICA given in Sec. 3 and Sec.4 for the much general cases given by eq.(5). *Furthermore, we can also prove the similar theorems as given by Theorems 1,2,3. In addition, we can also prove that the obtained* σ^{*2} *will be the true* σ_o^2 *for the algorithms in Sec.4* . Due to limited space, we leave these proofs to [14].

6. Bayesian Convex Divergence (BCYY) ICA Scheme

We go back to consider eq.(7b) by using the convex divergence eq.(3). Due to limited space, we leave the general case elsewhere, here we only consider the special case $n = m = n_o$ and $\sigma^2 = 0$. In this case, we have $y = s$ and $p_{M_1}(s|y) = \delta(s-y) = \delta(y-s) = p_{M_1}(y|s)$. Thus, eq.(7b) will becomes

$$\{W^*, \{g_i^*\}\} = arg\min_{\{W,\{g_i\}\in\mathcal{G}\}} J(W,\{g_i\}),$$

$$J(W,\{g_i\}) = -\int_x p_h(x) f(\frac{|W|\prod_{i=1}^n g_i(w_i x)}{p_h(x)})dx, \quad W = [w_1^T, \cdots, w_n^T]^T. \tag{27}$$

Through the density transformation $p(y) = p_h(x)/|W|$, we can also get

$$J(W,\{g_i\}) = -\int_y p(y) f(\frac{\prod_{i=1}^n g_i(y_i)}{p(y)})dy, \tag{28a}$$

which can be regarded as a generalized version of the MMI [3]. Moreover, we transform it further to $z = s(y)$ by a sigmoid monotonic function $s_i(r) = \int_{-\infty}^r g_i(y_i)dy_i$, resulting

$$J(W) = f(1) - \int_z p(z) f(\frac{1}{p(z)})dz. \tag{28b}$$

That is, we have reached a generalized version of the INFORMAX [4].

We consider the special case $f(u) = u^\beta$, $0 < \beta < 1$. From eq.(27), we have

$$J(W,\{g_i\}) = f(1) - \int_{x,} p_h^{1-\beta}(x)(|\det W|\prod_{i=1}^k g_i(x^t w_i))^\beta dx. \tag{29}$$

From $W\frac{\partial J(W,\{g_i\})}{\partial W}W^T$, we get both the batch way and adaptive algorithms for updating W by

$$\Delta W = -\eta\beta|\det W|^\beta\{E[p_h^{-\beta}(x)(\prod_{i=1}^k g_i(x^t w_i))^\beta(W + h(Wx)(Wx)^T)]W\},$$

$$\Delta W = -\beta|\det W|^\beta(\prod_{i=1}^k g_i(x^t w_i))^\beta(W + h(Wx)(Wx)^T W), \tag{30}$$

where h is the same as in eq.(10a). We have removed $p_h^{-\beta}(x)$ in the adaptive equation to save the computation on $p_h(x)$. It is interesting to compare eq.(30) with eq.(24b). We find that the moving step is modulated by a scalar $C(|\det W|, y, \beta) = \beta(|\det W|\prod_{i=1}^k g_i(x^t w_i))^\beta$. This change has one effect at least. Since $g_i(x^t w_i)$ is small when $|x^t w_i|$ is large, we have $C(|\det W|, y, \beta)$ becomes relatively small for large $|x^t w_i|$. In other words, the algorithm eq.(30) should be more robust to the affects by outliers.

To verify the effect, we use the fixed $h_i(s_i) = -s_i^3$ as in [10] which is shown that the algorithm eq.(24b) can success on sub-gaussian sources. In Fig.2, shown in the 1st row is the result of a problem of 2 channels with sources from sub-gaussian uniform(-1,1) signal plus 5% outliers. Shown in the 2nd row is the result of 2 channels with sources from sub-gaussian beta(0.5,0.5) signal plus 5% outliers. Shown in the 3rd row is the result of a problem of 3 channels with sources: one from the above uniform(-1,1) with outliers, one from the above beta(0.5,0.5) with outliers, and one from the super-gaussian permuted speech signal. Their histograms are listed in the 1st column from top down. Shown in the 2nd and 3rd columns are results by the algorithms eq.(30) and eq.(24b), respectively.

For the first two experiments (the first two rows), the cubic nonlinearity $h_i(s_i) = -s_i^3$ is used. The algorithm eq.(24b) given by [3,4] fails because the two channels of sources are actually super-gaussian now. The results are consistent to the existing theoretical results [15] that it cannot perform separation for super-gaussian signals. However, interestingly the algorithm eq.(30) with $\beta = 0.5$ successes. Trials with $\beta = 0.8, 0.2$ also show similar results. Hence, it is demonstrated that the algorithm eq.(30) is indeed more robust than

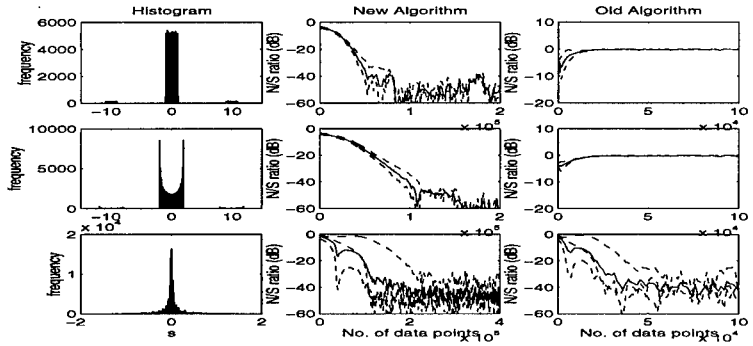


Figure 1 The experimental comparisons on several types of source signals

the algorithm eq.(24b) to outliers. The third experiment (the 3rd row) used the learned mixture of densities as used in [10,11] with $g_i(s_i)$ learned. Now both the BKYY-ICA and BCYY ICA algorithm work well.

References

[1] C.Jutten, "From source separation to Independent component analysis: An introduction to special session", Invited special session on Blind Signal Separation, Proc. of 1997 European Symp. on Artificial Neural Networks, Bruges, April 16-18, pp243-248.

[2] P. Comon, "Independent component analysis - a new concept?",*Signal Processing/* **36** (1994) 287-314.

[3] S.-I. Amari, A. Cichocki, H. Yang, " A new learning algorithm for blind separation of sources", in D. S. Touretzky, et al , eds, *Advances in NIPS 8*, MIT press, 1996 757-763.

[4] A.J. Bell and T.J. Sejnowski, " An information-maximatization approach to blind separation and blind deconvolution", *Neural Computation/* **7** (1995) 1129-1159.

[5] Xu, L., " Bayesian Ying-Yang System and Theory: An Unified Approach for Statistical Learning ", Invited paper, in S. Amari and N. Kassabov eds., *Brain-like Computing and Intelligent Information Systems*, 1997, Springer-Verlag.

[6] Xu, L, " New Advances on Bayesian Ying-Yang Learning System With Kullback and Non-Kullback Separation Functionals ", To appear on *Proc. of 1997 IEEE International Conference on Neural Networks (IEEE-INNS IJCNN97)*, June 9-12, 1997, Houston, TX, USA.

[7] Xu, L.,"Bayesian-Kullback YING-YANG Learning Scheme: Reviews and New Results", *progress in Neural Information Processing*, invited paper, *Proc. ICONIP96*, Sept. 24-27, 1996, pp59-67. Springer-verlag.

[8] Xu, L., "YING-YANG Machine: a Bayesian-Kullback scheme for unified learnings and new results on vector quantization", Keynote talk, Proc. Intl Conf. on Neural Information Processing (ICONIP95), Oct 30 - Nov. 3, 1995, pp977-988.

[9] Xu, L., and S. Amari, "A general independent component analysis framework based on Bayesian-Kullback Ying-Yang Learning", in *Progress in Neural Information Processing: Proc. ICONIP 96*, Hong Kong, 1996; Springer-Verlag, 1235-1239.

[10] Xu, L. C.C. Cheung, H.H. Yang and S. Amari, " Independent component analysis by the information-theoretic approach with Mixture of Density ", To appear on *Proc. of 1997 IEEE International Conference on Neural Networks (IEEE-INNS IJCNN97)*, June 9-12, 1997, Houston, TX, USA.

[11] Xu, L., C.C. Cheung, J. Ruan and S. Amari (1997b), "Nonlinearity and Separation Capability: Further Justification for the ICA Algorithm with A Learned Mixture of Parametric Densities", Invited special session on Blind Signal Separation, Proc. of 1997 European Symposium on Artificial Neural Networks, Bruges, April 16-18, pp291-296.

[12] Xu, L., "Bayesian-Kullback YING-YANG Machines for Supervised Learning", Invited Talk, Proceedings of 1996 World Congress On Neural Networks, San Diego, CA, Sept. 15-18, 1996, pp193-200.

[13] Xu, L., "Several New Results on Bayesian Ying-Yang Learning", Invited paper, to appear on *Proc. of Intl Workshop on Theoretical Aspects of Neural Computation,*, May 26-28, 1997 Hong Kong, Springer-Verlag.

[14] Xu, L., " Further Theoretical and Experimental Results on BYY ICA ", To be submitted to *Intl Conf. on Neural Information Processing (ICONIP97)*, Nov. 24-28, 1997, Zew Zealand.

[15] Cheung , C.C. and Xu, L , " Separation of Two Independent Sources by the Information-theoretic Approach with Cubic Nonlinearity", To appear on *Proc. of 1997 IEEE International Conference on Neural Networks (IEEE-INNS IJCNN97)*, June 9-12, 1997, Houston, TX, USA.

A NEURAL NETWORK APPROACH TO BLIND SOURCE SEPARATION *

Cristina Mejuto, Luis Castedo
Dpto. Electrónica y Sistemas, Universidad de La Coruña
Campus de Elviña s/n, 15.071 La Coruña, SPAIN
Tel: 3481 167150, Fax: 3481 167160, e-mail: luis@des.fi.udc.es

Abstract

The problem of adapting linear Multi-Input-Multi-Output systems for unsupervised separation of linear mixtures of sources arises in a number of signal processing applications. In this paper we present a new single layer neural network in which information transfer maximization is equivalent to minimizing a cost function involving the well-known Constant Modulus criterion originally used in blind equalization. The proposed approach is able to separate sources with negative kurtosis as revealed by an analysis of the cost function stationary points. Two learning rules are presented to compute the optimum separating matrix. One of them turns out to be an equivariant algorithm whose convergence does not depend on the mixture matrix.

1 Problem Statement

Adapting linear Multi-Input Multi-Output (MIMO) systems to separate linear mixtures of signals is a problem that frequently arises in signal processing applications such as array processing, multiuser detection, linear feature extraction, etc ... The blind source separation problem can be formulated as follows. Let us consider an array of sensors that provides a vector of observations $\mathbf{x} = [x_1, x_2, \cdots, x_N]^T$ which is a linear mixture of a vector of sources $\mathbf{s} = [s_1, s_2, \cdots, s_N]^T$

$$\mathbf{x} = \mathbf{A}\mathbf{s} \tag{1}$$

$\mathbf{A}$ represents the $N \times N$ mixture matrix. Both the sources and the mixture matrix are unknown. The only assumptions we will make in our model are $\mathbf{A}$ is a non-singular matrix and the sources are zero-mean, statistically independent, non-gaussian random processes. Without loss of generality, we

*This work has been supported by Xunta de Galicia (grant XUGA 10502A96) and CICYT (grant TIC 96-0500-C10-02)

0-7803-4256-9/97/$10.00 ©1997 IEEE

can assume that sources have unit variance since power differences can be incorporated in matrix $\mathbf{A}$.

To recover the sources, $\mathbf{x}$ is processed through a linear memoryless MIMO system, represented by a $N \times N$ matrix $\mathbf{W}$, to produce an output vector $\mathbf{y} = [y_1, y_2, \cdots, y_N]^T$

$$\mathbf{y} = \mathbf{W}^T\mathbf{x} \tag{2}$$

The superindex ^{T} denotes transpose. Combining both (1) and (2) together, we get

$$\mathbf{y} = \mathbf{G}\mathbf{s} \tag{3}$$

where $\mathbf{G} = \mathbf{W}^T\mathbf{A}$ is the matrix representing the overall mixing/separating system. The objective in source separation is to select $\mathbf{W}$ in order that each output corresponds to a single and different source s_i up to some fixed gain. When this occurs, $\mathbf{G}$ can be expressed as the product of a diagonal matrix $\mathbf{\Delta}$ and a permutation matrix $\mathbf{P}$, i.e., $\mathbf{G} = \mathbf{\Delta}\mathbf{P}$.

A basic principle to solve the blind source separation problem is provided by the Darmois-Skitovich theorem [1]: if $\mathbf{s}$ is a vector of statistically independent non-gaussian signals and $\mathbf{y} = \mathbf{G}\mathbf{s}$, $\mathbf{y}$ is a vector of statistically independent signals if and only if $\mathbf{G} = \mathbf{\Delta}\mathbf{P}$. Therefore, one way to recover the sources is to select $\mathbf{W}$ in order to minimize the statistical dependence among the components in $\mathbf{y}$. This is referred in the literature as Independent Component Analysis (ICA) [2] and a number of both block processing [2] and adaptive processing [3, 4, 5, 6] methods have been proposed. In the sequel, we will focus our attention into adaptive methods since they are easier to implement and more related to the field of neural networks.

Recently, several adaptive algorithms for blind source separation have been developed in the context of unsupervised learning of neural networks [7]. The separating MIMO system is then interpreted as the linear part of a single layer nonlinear neural network (see figure 1). In this model, matrix $\mathbf{W}$ represents the synaptic weights and $g(\cdot)$ the activation function. The vector of the outputs after the nonlinearities $u_i = g(y_i),\ i = 1, \cdots, N$ is denoted $\mathbf{u}$.

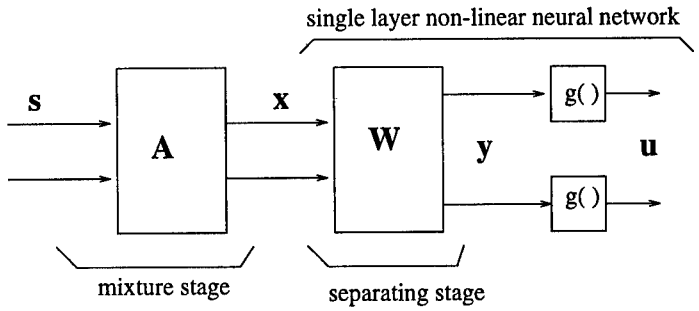


Figure 1: The model.

Trying to understand the way perceptual systems work, an unsupervised

learning paradigm called the infomax principle [8] has been proposed. According to this principle the parameters of a neural network should be chosen to maximize the information transfer between the input and the output. Nadal and Parga [9] have shown that if the activation functions are continuous, increasing, invertible and bounded nonlinearities, information transfer between $\mathbf{x}$ and $\mathbf{u}$ is maximized when the components in $\mathbf{u}$ are statistically independent and have a uniform distribution. This result suggests that statistical dependence between outputs tends to reduce when maximizing information transfer. However, this is not always true since simulations reported in [7] show that information transfer maximization algorithms only perform blind source separation when the sources have positive kurtosis. In many applications (communications and image processing, for instance) signals typically have negative kurtosis and the algorithms in [7] do not work adequately.

In this paper we present a new information transfer maximization algorithm suitable for sources with negative kurtosis. The algorithm can also be interpreted as a generalization of the Constant Modulus Algorithm (CMA) [10], therefore reinforcing the link between information-theoretic unsupervised learning paradigms and blind adaptive filtering. Section 2 presents the information transfer maximization criterion. Section 3 introduces two learning rules. Section 4 presents some simulation experiments and section 5 contains the conclusions.

2 Information Transfer Maximization

Let us start by considering a single layer nonlinear neural network comprising a linear part and a new fixed activation function defined as follows

$$g(x) = \int_{-\infty}^{x} exp(-(t^2-1)^2)dt \tag{4}$$

Figure 2 shows the plot of $g(x)$ and, similarly to other well-known activation functions, it is a continuous, increasing, bounded and invertible nonlinearity.

Following the infomax principle [8], we select the synaptic weights $\mathbf{W}$ in order to maximize the information transfer between the input $\mathbf{x}$ and the output after the nonlinearity $\mathbf{u}$, that is

$$I(\mathbf{x},\mathbf{u}) = E\left\{\ln \frac{p_{x,u}(\mathbf{x},\mathbf{u})}{p_x(\mathbf{x})p_u(\mathbf{u})}\right\} \tag{5}$$

where $E\{\cdot\}$ denotes expectation, and $p_x(\mathbf{x})$, $p_u(\mathbf{u})$ and $p_{x,u}(\mathbf{x},\mathbf{u})$ are the probability density functions (p.d.f.) of $\mathbf{x}$, $\mathbf{u}$ and the pair $(\mathbf{x},\mathbf{u})$ respectively. Taking into account that the entropy of the output $\mathbf{u}$ is $H(\mathbf{u}) = E\{-\ln p_u(\mathbf{u})\}$ and that the entropy of $\mathbf{u}$ conditioned to the input $\mathbf{x}$ is $H(\mathbf{u}|\mathbf{x}) = E\{-\ln p_{u,x}(\mathbf{u}|\mathbf{x})\}$, (5) is equivalent to

$$I(\mathbf{x},\mathbf{u}) = H(\mathbf{u}) - H(\mathbf{u}|\mathbf{x}) \tag{6}$$

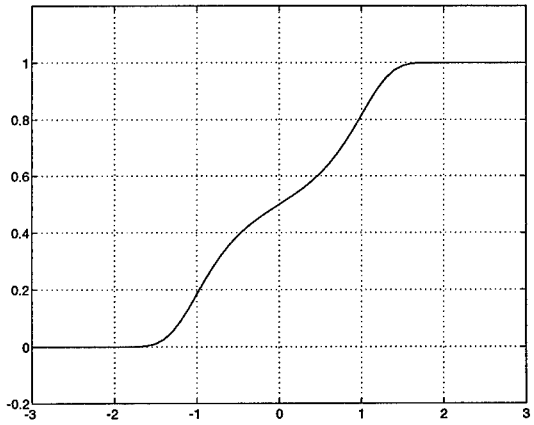


Figure 2: Nonlinear transfer function.

Now, since the relationship between $\mathbf{u}$ and $\mathbf{x}$ is deterministic, $H(\mathbf{u}|\mathbf{x}) = 0$. Therefore, maximizing the information transfer is equivalent to maximizing the output entropy $H(\mathbf{u})$ [9].

Next, we will express $H(\mathbf{u})$ in terms of the input entropy. Provided that $\mathbf{W}$ is a square and invertible matrix and the nonlinearity in (4) is invertible, $\mathbf{u}$ has a p.d.f. given by

$$p_u(\mathbf{u}) = \frac{p_x(\mathbf{x})}{|det\ \mathbf{W}| \prod_{i=1}^{n} g'(y_i)} \tag{7}$$

where $\mathbf{y} = \mathbf{W}^T\mathbf{x}$. Therefore,

$$H(\mathbf{u}) = H(\mathbf{x}) + \sum_{i=1}^{N} E\{\ln\ g'(y_i)\} + \ln\ |det\ \mathbf{W}| \tag{8}$$

where $H(\mathbf{x})$ is the input entropy. Particularizing for the nonlinearity (4)

$$H(\mathbf{u}) = H(\mathbf{x}) - \sum_{i=1}^{N} E\left\{\left(y_i^2 - 1\right)^2\right\} + \ln\ |det\ \mathbf{W}| \tag{9}$$

Since the input entropy $H(\mathbf{x})$ does not depend on $\mathbf{W}$, we conclude that maximizing $H(\mathbf{u})$ (or equivalently maximizing the information transfer) is equivalent to minimizing the cost function

$$\phi(\mathbf{W}) = \sum_{i=1}^{N} E\{(y_i^2 - 1)^2\} - \ln|det\ \mathbf{W}| \tag{10}$$

This is an important result because this cost function also admits another interesting interpretation from the perspective of blind adaptive filtering.

The first part of (10) is the extension to MIMO systems of the well-known Constant Modulus (CM) criterion [10] widely used in blind equalization. The analysis carried out in [11] for a Multiple-Input-Single-Output (MISO) system whose output is $y_i = \mathbf{w}_i^T \mathbf{x}$ shows that if the kurtosis of all the sources is negative, the only existing minima of the criterion $E\{(y_i^2 - 1)^2\}$ correspond to points where a single source is extracted. This means that if a MIMO system is adjusted according to the first part of (10), each output y_i will extract a single source. However, there exists the possibility that the same source is extracted at different outputs. This situation is prevented by the existence of the second term in (10) because, when it occurs, two columns of $\mathbf{W}$ are proportional and the second part of (10) grows very large.

The ability of our approach to perform source separation is further supported by an analysis of the stationary points of $\phi(\mathbf{W})$ presented in [12]. A simple situation of a two sources mixture and a two-inputs-two-outputs neural network was assumed. The analysis consisted in finding the points where the gradient vanishes and examining the positive definiteness of the Hessian matrix at these points to determine whether they are maxima, minima or saddle points. It was possible to prove that the points $\mathbf{W}$ where source separation is achieved correspond to minima if the kurtosis of the sources is negative. The analysis turned rather involved when trying to show that $\phi(\mathbf{W})$ does not contain undesirable stationary points. Nevertheless, computer simulations never revealed such undesirable equilibria points.

3 Learning Rules

In this section we discuss different learning rules to compute the coefficients of the separating matrix that minimize $\phi(\mathbf{W})$. The first possibility is to use a gradient descent algorithm of the form

$$\mathbf{W}(n+1) = \mathbf{W}(n) - \mu \frac{\partial \phi}{\partial \mathbf{W}(n)} \tag{11}$$

where μ is the algorithm step-size. Taking (2) into account and that $det\ \mathbf{W} = \sum_{j=1}^{N} w_{ij}\ cofw_{ij}$ for any row i ($cofw_{ij}$ being the cofactor of w_{ij}) we have

$$\frac{\partial \phi}{\partial w_{ij}} = 4E\{x_i y_j^3\} - 4E\{x_i y_j\} - \frac{cofw_{ij}}{det\ \mathbf{W}} \tag{12}$$

Dropping the expectation operator, the resulting stochastic gradient descent algorithm reads

$$\mathbf{W}(n+1) = \mathbf{W}(n) + \mu \left(4\ \mathbf{x}\mathbf{y}^T\ -\ 4\ \mathbf{x}\mathbf{y}^T\mathbf{D}(\mathbf{y})\ +\ \left[\mathbf{W}^T(n)\right]^{-1}\right) \tag{13}$$

where $\mathbf{D}(\mathbf{y}) = Diag[y_1^2, y_2^2, \cdots, y_N^2]$. As discussed in [11], the term $+\mathbf{x}\mathbf{y}^T$ in (13) is a typical anti-Hebbian term. The term $-\mathbf{x}\mathbf{y}^T\mathbf{D}(\mathbf{y})$ has the twofold effect of stabilizing the algorithm and incorporating high order statistics to

reach output independence (it is well-known that the anti-Hebbian rule is by itself unstable and reaches output decorrelation, not independence). It constitutes an improved version of the adaptive Oja's algorithm [13] for principal component analysis. Finally, the term $\left[\mathbf{W}^T\right]^{-1}$ precludes convergence towards a singular matrix $\mathbf{W}$.

In order to avoid computation of an inverse matrix at each update, a second choice as a learning rule is a relative gradient descent algorithm [5] of the form

$$\mathbf{W}(n+1) = \mathbf{W}(n) - \mu \mathbf{W}(n)\mathbf{W}^T(n)\frac{\partial \phi}{\partial \mathbf{W}(n)} \tag{14}$$

As long as $\mathbf{W}(n)$ remains a nonsingular matrix, it is easily shown that this form of adaptation always reduces $\phi(\mathbf{W})$ [5]. From (13) it is apparent that the resulting stochastic relative gradient algorithm is

$$\mathbf{W}(n+1) = \mathbf{W}(n) + \mu \mathbf{W}(n)\left(4\ \mathbf{y}\mathbf{y}^T\ -\ 4\ \mathbf{y}\mathbf{y}^T\mathbf{D}(\mathbf{y})\ +\ \mathbf{I}\right) \tag{15}$$

where $\mathbf{I}$ is the identity matrix. Again, each term in (15) has a clear interpretation from the perspective of source separation. The term $+\mathbf{y}\mathbf{y}^T$ forces decorrelation between the different outputs whereas the second term $-\mathbf{y}\mathbf{y}^T\mathbf{D}(\mathbf{y}$ involves higher order statistics and forces the stronger condition of independence between different outputs. Finally, the identity matrix $\mathbf{I}$ prevents the algorithm to converge towards a solution where all the outputs are equal to zero.

Although motivated by the necessity of avoiding matrix inverse computations, the learning rule (15) exhibits the more interesting property of being an equivariant algorithm [5]. Premultiplying (15) by $\mathbf{A}^T$ it is obtained that the combined mixing/separating system evolves under the following updating rule

$$\mathbf{G}^T(n+1) = \mathbf{G}^T(n) + \mu \mathbf{G}^T(n)\left(4\ \mathbf{y}\mathbf{y}^T\ -\ 4\ \mathbf{y}\mathbf{y}^T\mathbf{D}(\mathbf{y})\ +\ \mathbf{I}\right) \tag{16}$$

that does not depend explicitly on the mixing matrix $\mathbf{A}$. As a consequence, (15) possess the equivariance property because the time evolution of the global system is independent of $\mathbf{A}$: it only depends on the initial conditions and the statistical characteristics of the sources $\mathbf{s}$. It will perform adequately even though $\mathbf{A}$ is an ill-conditioned matrix.

4 Computer Simulations

In this section we present the results of some computer experiments carried out to illustrate the performance of the equivariant learning rule (15) and establish comparisons with existing approaches. As sources, we consider three images with 256×256 pixels having a normalized kurtosis of -1.42, -0.75 and -0.51. These images can be seen in the left column of figure 3.

In the first simulation experiment we considered the following mixture matrix

$$\mathbf{A} = \begin{bmatrix} -1 & -1 & 1 \\ 1 & 1 & 1 \\ 1 & -1 & 1 \end{bmatrix} \tag{17}$$

The resulting mixed observations are plotted in the middle column of figure 3. To recover the sources from the observations, a 3×3 MIMO system is considered whose coefficients are updated according to the equivariant algorithm (15). To reduce the misadjustment noise, we chose a variable step-size that starts from $\mu = 5 \times 10^{-4}$ and is multiplied by $0.6 \cdot$ each 10.000 iterations. The right column of figure 3 shows the outputs corresponding to the separating system obtained after 65.536 iterations which is the size of a 256×256 image. It is clearly seen that our approach was able to successfully recover the original sources.

In order to measure the performance of our algorithm and make comparisons with existing approaches we define the following index [11] which is zero iff $\mathbf{G} = \mathbf{W}^T\mathbf{A}$ corresponds to source separation

$$\rho(\mathbf{W}) = \sum_{i=1}^{N} \left(\sum_{j=1}^{N} \frac{g_{ij}^2}{max_l g_{il}^2} - 1 \right) + \sum_{j=1}^{N} \left(\sum_{i=1}^{N} \frac{g_{ij}^2}{max_l g_{lj}^2} - 1 \right) \tag{18}$$

Figure 4 plots the time evolution of this performance index for the proposed algorithm (15), the EASI algorithm with a cubic nonlinearity [5], Bell and Sejnowski [7] (BS) and Cichocki and Unbehauen [4]. These three latter were implemented with a variable step-size strategy similar to the one described above. It is apparent that our approach performs almost the same as the EASI algorithm whereas outperforms BS and CU.

Finally, to test the equivariance property of the learning rule (15), we carried out a second simulation experiment considering the ill-conditioned mixture matrix

$$\mathbf{A} = \begin{bmatrix} 1.01 & 1 & 1 \\ 1 & 1.01 & 1 \\ 1 & 1 & 1.01 \end{bmatrix} \tag{19}$$

Figure 5 plots the performance index for the same algorithms as before showing again that the EASI algorithm and ours exhibit superior performance than BS and CU. In addition, the speed of convergence has remained unchanged.

5 Conclusions

Existing information transfer maximization algorithms [7] that use conventional activation functions are only capable of separating sources with positive kurtosis and therefore cannot be used in communications or image processing applications where signals typically have negative kurtosis. This paper overcomes this limitation by presenting a single layer neural network with a new

activation function. It is shown that maximizing its information transfer is equivalent to minimizing a statistical criterion that involves the well-known Constant Modulus criterion [10] originally used for blind equalization. Two learning rules have been derived, a conventional gradient descent rule and a relative or natural gradient descent rule. The latter turns out to be an equivariant algorithm whose performance is independent of the mixture matrix. Finally, simulations show that our approach performs the same or better than existing blind source separation adaptive algorithms when applied to sources with negative kurtosis.

References

[1] X. R. Cao, R. W. Liu, "General Approach to Blind Source Separation", *IEEE Trans. Signal Processing*, vol. SP-44, pp. 562-571, March 1996.

[2] P. Comon, "Independent Component Analysis, A New Concept?", *Signal Processing*, vol. 36, pp. 287-314, April 1994.

[3] E. Moreau, O. Macchi, "High-Order Contrasts for Self-Adaptive Source Separation", *International Journal of Adaptive Control and Signal Processing*, vol. 10, pp. 19-46, 1996.

[4] A. Cichocki, R. Unbehauen, "Robust Neural Networks with On-Line Learning for Blind Indentification and Blind Separation of Sources", *IEEE Trans. Circuits and Systems-I*, vol. 43, pp. 894-906, Nov. 1996.

[5] J. F. Cardoso and B. Laheld, "Equivariant Adaptive Source Separation", *IEEE Trans. Signal Processing*, vol. SP-44, pp. 3017-3030, Dec. 1996.

[6] L. Castedo, C. Escudero, A. Dapena, "A Blind Signal Separation Method for Multiuser Communications", to appear in *IEEE Trans. Signal Processing*, May 1997.

[7] A. J. Bell, T. J. Sejnowski, "An Information–Maximization Approach to Blind Separation and Blind Deconvolution", *Neural Computation*, vol. 7, pp. 1129-1159, Nov. 1995.

[8] S. Haykin, *Neural Networks, A Comprehensive Foundation*, Macmillan College Publishing Company, Inc., 1996

[9] J. P. Nadal and N. Parga, "Non Linear Neurons in the Low Noise Limit: a Factorial Code Maximizes Information Transfer", *Network*, vol. 5, pp. 565-581, 1994.

[10] D. N. Godard, "Self-Recovering Equalization and Carrier Tracking in Two-Dimensional Data Communication Systems", *IEEE Trans. on Communications*, vol. COM-28, pp. 1867-1875, November 1980.

[11] Z. Malouche and O. Macchi, "Adaptive Unsupervised Extraction of One Component of a Linear Mixture with a Single Neuron", *submitted to IEEE Trans. Neural Networks*, 1996.

[12] L. Castedo, O. Macchi, "Maximizing the Information Transfer for Adaptive Unsupervised Source Separation", *Proc. IEEE Workshop on Signal Processing Advances in Wireless Communications SPAWC'97*, Paris, France, pp. 65-68, April 1997,

[13] E. Oja, "Principal Components, Minor Components, and Linear Neural Networks", *Neural Networks*, vol. 5, pp. 927-935, 1992.

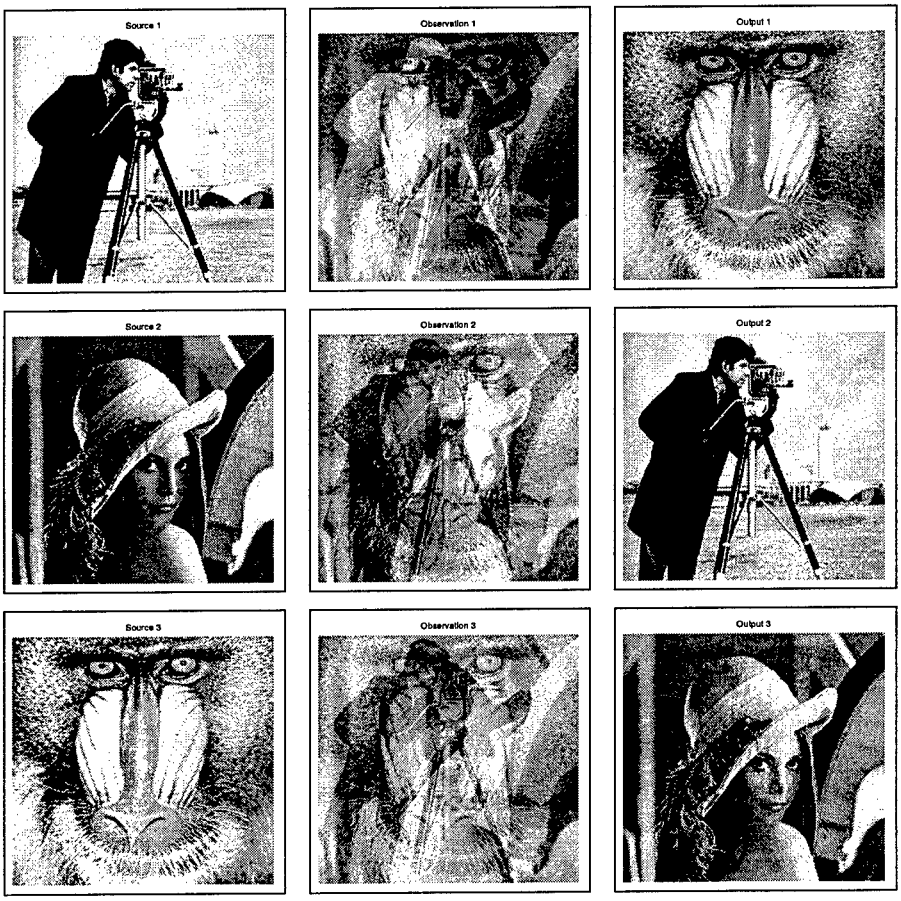


Figure 3: Blind separation of three images with the proposed algorithm.

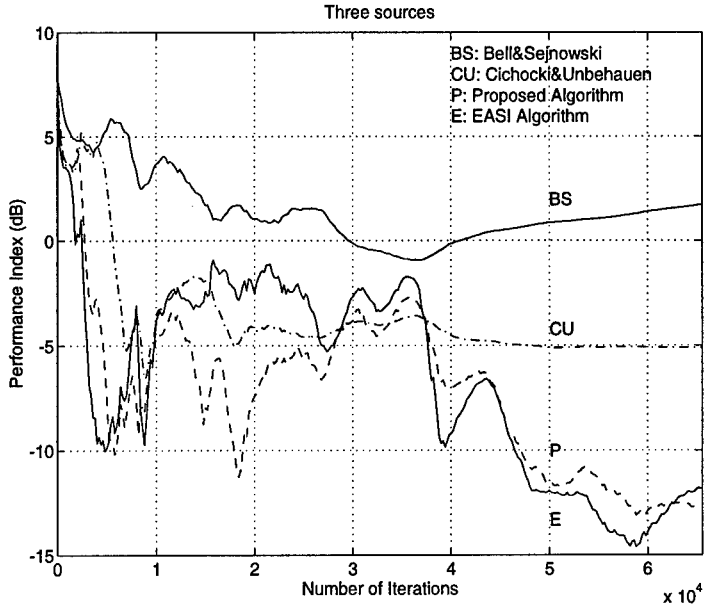


Figure 4: Performance index for the first computer experiment with the well-conditioned mixture matrix (17).

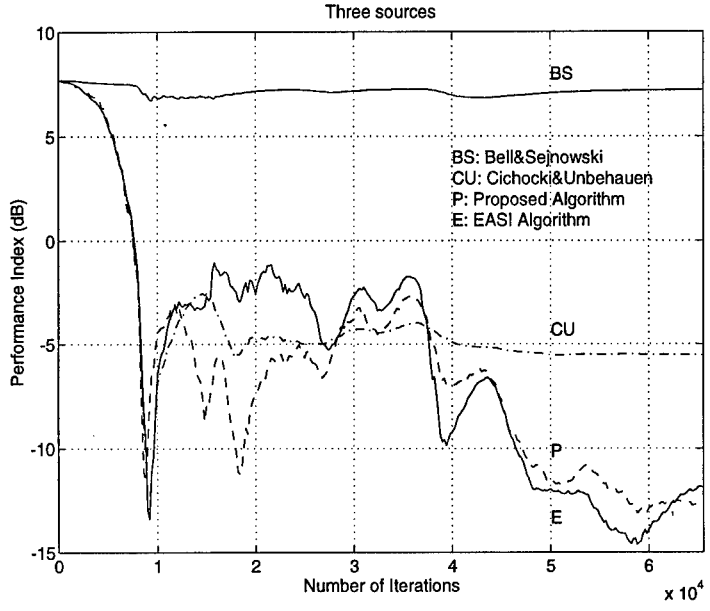


Figure 5: Performance index for the second computer experiment with the ill-conditioned mixture matrix (19).

A Unifying Criterion for Blind Source Separation and Decorrelation: Simultaneous Diagonalization of Correlation Matrices

Hsiao-Chun Wu, Jose C. Principe

Computational Neuro-Engineering Laboratory
Department of Electrical and Computer Engineering
University of Florida, Gainesville, FL 32611
{principe, wu}@cnel.ufl.edu

Abstract

Blind source separation and blind output decorrelation are two well-known problems in signal processing. For instantaneous mixtures, blind source separation is equivalent to a generalized eigen-decomposition, while blind output decorrelation can be considered as an iterative method of output orthogonalization. We propose a steepest descent procedure on a new cost function based on the Frobenius norm which measures the diagonalization of correlation matrices to perform blind source separation as well as blind decorrelation. The method is applicable to both stationary and nonstationary signals and instantaneous as well as convolutive mixture models. Simulation results by Monte Carlo trials are provided to show the consistent performance of the proposed algorithm.

1. Introduction

The field of blind signal processing which includes blind source separation, blind decorrelation and blind equalization has recently received a lot of attention. Most of the blind separation algorithms are based on high-order statistical information because it can be shown that second order statistics are not sufficient to uniquely separate sources [15]. However, this proof requires signal stationarity which is not applicable to many important real life problems (such as separation of speech). There have been also reports showing experimentally that systems based on second order statistics can indeed separate mixed sources [1]-[9]. These methods explore the time characteristics of the covariance, i.e. either nonstationary temporal estimates of covariance matrices or time-delayed cross-correlation matrices. Blind decorrelation can be formulated with second order statistics and can be solved by the orthogonalization of covariance matrices [11, 12]. Therefore methods based on second order statistics for blind source separation are related to blind decorrelation, but there is no systematic coverage of the two areas. In this paper, we will unify the

0-7803-4256-9/97/$10.00 ©1997 IEEE

framework of blind source separation using second order statistics with blind decorrelation and apply the new algorithm to speech data.

The signal separation and decorrelation problems can be modeled as

$$X(t) = \left[x_k(t)\right] = \left[\sum_i a_{ki} s_i(t)\right] = AS(t) \qquad (1)$$

where S(t) is an N by 1 zero-mean vector containing the original signal, X(t) is an M by 1 coupled signal vector ($M >= N$), A is an M by N real time-invariant coupling matrix, s_i(t) is the i^{th} unknown signal and a_{ki} is a real unknown coupling coefficient. If A is full rank, we can always use PCA to find the subspace of X(t) which is equivalent to the space of S(t). The objective of blind decorrelation is to design a full-rank weight matrix W that constructs an output $Y(t) = W^T X(t)$ displaying a diagonal output covariance $E\{Y(t)Y^T(t)\}$. However, blind separation has a different objective since one needs to design a signal separator W which is able to reconstruct the original signals. The latter constraint will put a stronger requirement on the weight matrix W such that

$$W^T = A^{-1} \qquad \text{or} \qquad W^T A = PD \qquad (2)$$

where P is a permutation matrix, D is a diagonal scaling matrix and T denotes matrix transpose [13]. The weight matrices for blind separation belong to a subset of the blind decorrelation solution.

2. Approach and criterion

According to [4, 5], blind separation of nonstationary signals can be formulated as the simultaneous diagonalization of two covariance matrices estimated at different times, which can be further reduced to a generalized eigen-decomposition problem (requiring off-line processing). Orthogonalization of covariance estimates at many time instants with regularization was suggested in [6] as an on-line algorithm for blind separation. However, the method can only decorrelate signals with positive-definite covariance matrices due to a restriction placed on the cost function. Instead of using different covariance estimates at different time intervals, one can still perform blind separation (decorrelation) through the simultaneous orthogonalization of two or more time-delayed correlation matrices [7, 8] as

$$W^T E\{X(t)X^T(t-q)\}W = D(q) \qquad (3)$$

where D(q) is a diagonal matrix associated with a delay q. The non-symmetrical time delayed correlation matrix $E\{X(t)X^T(t-q)\}$ is not necessarily positive-definite and hence we cannot apply the criterion proposed by [6]. To our knowledge only

the generalized eigen-decomposition was proposed to solve this formulation but the method is an analytic solution [7, 8]. Here we will propose an alternative criterion to solve the realistic case of non-positive-definite time-delayed correlation matrices iteratively.

The idea is to create a cost function which minimizes directly the difference between the quadratic form in the left side of Eq. (3) and its right side. In order to measure the distance between the correlation estimate $E\{X(t)X^T(t-q)\}$ and its diagonal version, $D_q(t)$ we propose the following criterion:

$$\sum_{q=0}^{d} J_q(t) = \sum_{q=0}^{d} \left\| W^T E\{X(t)X^T(t-q)\} W - D_q(t) \right\|_F^2 \tag{4}$$

$$\text{as well as } J_q(t) = \left\| W^T E\{X(t)X^T(t-q)\} W - D_q(t) \right\|_F^2 \tag{5}$$

where $\| \ \|_F$ denotes the *Frobenius norm* and d is the total number of the delayed covariance matrices $D_q(t)$ we need to constrain. However the choice of d still needs to be further analyzed. The Frobenius norm is defined as [14]:

$$\|A\|_F = \sqrt{\sum_{i}\sum_{j} |a_{ij}|^2} \tag{6}$$

where A is an m-by-n matrix. The minimum of this cost function preserves the diagonal elements of $E\{X(t)X^T(t-q)\}$ but zeros all other elements out, i.e., it solves both the blind separation and decorrelation problems. The cost function of Equation (4) is a nonnegative fourth-order function of the weight coefficients W. The minimum can be obtained using a gradient descent procedure $\Delta W = -\eta \sum_{q=0}^{d} \frac{dJ_q(t)}{dW}$. Simple linear algebra manipulations yield

$$\Delta W = -\eta \sum_{q=0}^{d} \{[C_q(t) + C_q^T(t)] W [W^T C_q(t) W - D_q(t)]\} \tag{7}$$

$$\text{where} \qquad C_q(t) = E\{X(t)X^T(t-q)\} \tag{8}$$

In addition, the gradient descent procedure using our proposed criterion will move the directions of the output signals in such a way that orthogonalizes all the vectors in tandem as depicted in Figure 1. This methodology is different from Gram-Schmidt where the principal component must stabilize before the others converge since they are corrected with respect to it. (yielding what is called the deflation pro-

cedure.) Deflation procedures converge sequentially which is a known problem for the recent prevalent orthogonalization procedures based on the L2 norm or Rayleigh quotient optimization.

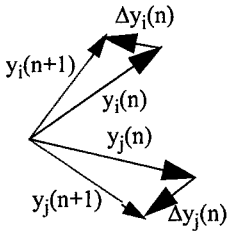


Figure 1. Illustration of orthogonalization procedure

Our procedure does not constrain the size of the vectors, so null vectors can occur during learning. This corresponds to a trivial solution that meets the minimization of the criterion of Eq. (4). To avoid it, we have to impose a unit length constraint on the weight vectors.

3. Implications of the new criterion

For the case $D_q(t) = I$ and q = 0, Eq. (7) yields,

$$W(t+1) = W(t) - \eta W(t) E\{Y(t)Y^T(t)\}[E\{Y(t)Y^T(t)\} - I] \quad (9)$$

We will show that this adaptation rule was previously utilized in blind decorrelation and independent component analysis.

3.1 Stochastic Whitening Procedure

If the requirement is to obtain whitened outputs, we have to estimate $E\{Y(t)Y^T(t)\}$ either using an exponential window or batch mode. In [12] the following on-line stochastic whitening procedure was derived through a Gram-Schmidt-like orthogonalization,

$$W(t+1) = W(t) - \eta W(t)[Y(t)Y^T(t) - I] \quad (10)$$

which is equivalent to Eq. (9) if the first $E\{y(t)y^T(t)\}$ is dropped.

3.2 Kullback-Leibler Divergence

If we replace $Y(t)$ by $f(Y(t))$ in Eq. (10), where $f(*)$ is a proper nonlinear function, we obtain the update rule derived from minimization of Kullback-Leibler divergence in [10].

3. 3 Generalized Blind Decorrelation Rule

The adaptation rule for generalized decorrelation at iteration n using Eq. (7) with an arbitrary square matrix $C_q(t)$ is

$$W_{n+1} = W_n + \Delta W_n \quad \text{subject to } w^T_n(i)w_n(i) = 1, \text{ for all } i = 1, 2, ..., N. \qquad (11)$$

where $w_n(i)$ is the i^{th} column vector of W_n. With Eq. (11), we can extend the solution presented in [11] to any time-delayed correlation matrix. If we orthogonalize several time-delayed matrices using this procedure, we can obtain separated signals as in [7].

3.4 On-line Adaptation Rule for Blind Separation of Nonstationary Signals (Instantaneous mixture)

It has been proved in [6] that, for linear time-invariant instantaneous mixture of locally stationary signals, the source signals can uniquely be determined from the sensed signals (except the arbitrariness of the permutation matrix P and the diagonal matrix D) if and only if Eq. (12) holds

$$E\{y_i(t)y_j(t)\} = 0 \quad \text{for all} \quad i \neq j \quad \text{at any instant of time t.} \qquad (12)$$

Eq. (12) can be easily translated as the orthogonalization of the output correlation matrix at any instant of time t as in Eq. (13)

$$W^T E\{X(t)X^T(t)\}W = D_0(t) \quad \text{at any instant of time t.} \qquad (13)$$

Here we propose to use batch learning with non-overlap windowed estimates. The blind source separation rule for nonstationary signals becomes

for the m^{th} batch

$W_{m+1} = W_m + \Delta W_m$, subject to $w^T_m(i)w_m(i) = 1$, for all $i = 1, 2, ..., N$.

where $w_m(i)$ is the i^{th} column vector of W_m.

$C_0(m)$ has to be used in Eq. (7) to calculate ΔW_m.

4. Blind source separation of linear convolutive mixture

If we have the sensed signal $X(t) = [x_1(t)\, x_2(t), ..., x_N(t)]^T$, composed by a linear

convolutive mixture of sources, $X(z) = H(z)S(z)$, where

$$H(z) = \begin{bmatrix} h_{11}(z) & \cdots & h_{1N}(z) \\ \cdots & h_{ij}(z) & \cdots \\ h_{N1}(z) & \cdots & h_{NN}(z) \end{bmatrix} \qquad S(z) = \begin{bmatrix} s_1(z) \\ \cdots \\ s_N(z) \end{bmatrix}$$

The solution $Y(z)$ for separated signals is $Y(z) = PK(z)H^{-1}(z)$ where P is a permutation matrix and $K(z)$ is a diagonal matrix with some arbitrary shaping filters as its diagonal elements. We can modify Eq. (8) as

$$C_q(t) = E\{\vec{X}(t)\vec{X}(t-q)\} \tag{14}$$

where $\vec{X}(t)$ is

$$\vec{X}(t) = [\vec{x}_1\vec{x}_2\ldots\vec{x}_N]^T;\ \vec{x}_j = [x_j(t)x_j(t-1)\ldots x_j(t-L+1)]_{1\le j\le N}. \tag{15}$$

In addition, we also need to modify the separation weight matrix W as

$$W = \begin{bmatrix} \vec{w}_{11} & \vec{w}_{21} & \cdots & \vec{w}_{N1} \\ \vec{w}_{12} & \vec{w}_{22} & \cdots & \vec{w}_{N2} \\ \cdots & \cdots & \vec{w}_{ij} & \cdots \\ \vec{w}_{1N} & \vec{w}_{2N} & \cdots & \vec{w}_{NN} \end{bmatrix} \tag{16}$$

where $\vec{w}_{ij} = [w_{ij}(0), w_{ij}(1), \ldots, w_{ij}(L-1)]^T_{1\le i,j\le N}$ is the separating filter of length L. With the new definitions of Eq. (14), (15) and (16), we can apply Eq. (7) to solve for linear convolutive mixture.

5. Simulation results

Since the solution for blind source separation is a subset of blindly decorrelated outputs we focus our first experiment on blind source separation of instantaneous mixed signals. We select speech signals spoken by two male speakers (TIMIT database), and we artificially mix them with a matrix A. We acknowledge that this a simplified problem, but it is one where we have control of the experiments to test the new algorithm. The method can be easily extended to any N by N case.

We choose the mixing matrix A randomly and average the performance by sixty *Monte Carlo* trials with sixty different random initial weights. Then we run the experiments for 200 epochs according to our proposed adaptation rule. We seek with this experiment to find how reliable is the method, i.e. how many times the

solution is found and what is the variance in the estimates. Figure 2 and 3 plot the mean Frobenius distances given by Eq. 4 versus epoch number with the corresponding one standard deviation errorbars (upper / lower limits of performance curves). In Figure 2 the criterion is J_0 and all the Monte Carlo runs converged to the true solution in less than 20 epochs. On the other hand, Figure 3 shows the fast convergence of our algorithm when we try to minimize the combined criterion $\sum_{q=0}^{3} J_q$. Among the trials we select one to illustrate the convergence results. The particular instantaneous mixture matrix was

$$A = \begin{bmatrix} 0.7012 & 0.7622 \\ 0.9103 & 0.2625 \end{bmatrix} \tag{17}$$

After only 4 to 12 learning epochs, the system reaches the optimization with the separation weight matrix W_{opt}. We can investigate if the product of two matrices $W^T A$ is actually in the form of PD as previously described [13]. Consequently we normalize each row of the product matrix $W^T_{opt} A$ by the absolute value of its dominant element and the product as

$$W^T_{opt} A = \begin{bmatrix} -0.0025 & -1.0000 \\ 1.0000 & 0.0055 \end{bmatrix} \tag{18}$$

From Eq. (15) we can see that we have really removed almost all the interference from the other source in this trial picked at random. From Eq. (18) we can see that *SNR* (signal-to-noise ratio) is 52.04 dB at receiver 1 and 45.19 dB at receiver 2 respectively. The distortion is almost unnoticeable.

In our second experiment, for a convolutive mixture of two sources, we choose an arbitrary mixing matrix

$$H(z) = \begin{bmatrix} 1 & 0.85z^{-2} + 0.1z^{-3} \\ 0.7z^{-1} + 0.4z^{-2} + 0.25z^{-3} & 1 \end{bmatrix} \tag{19}$$

The algorithm needs approximately 150 epochs to converge as shown by the learning curve plotted in Figure 4 (L = 6 in Eq. (15), d = 20 in Eq. (4) and the energy of $s_1(t)$ and $s_2(t)$ are equal for this case). Since the convolutive model is more complicated we cannot simply apply the product in Eq. (18) to investigate the simulation result. However we can compute the output as

$$Y(z) = W^T(z)H(z)S(z) = \begin{bmatrix} \varpi_{11}(z) & \cdots & \varpi_{N1}(z) \\ \cdots & \varpi_{ij}(z) & \cdots \\ \varpi_{1N}(z) & \cdots & \varpi_{NN}(z) \end{bmatrix}^T H(z)S(z). \tag{20}$$

where $\varpi_{ij}(z)$ is the z-transform of $\vec{w}$ for $1 \le i, j \le N$, and $H(z)$, $S(z)$ were previously defined. Hence from Eq. (20) the outputs for the simulation of the convolutive model can be obtained as

$$Y_i(z) = \alpha_{i1}(z)S_1(z) + \alpha_{i2}(z)S_2(z) \text{ for } i = 1, 2. \tag{21}$$

or $y_i(t) = y_{i1}(t) + y_{i2}(t)$ where $y_{ij}(t)$ is the inverse z-transform of $\alpha_{ij}(z)S_j(z)$.

According to Eq. (21) the energy matrix can be defined as [8]

$$\begin{bmatrix} \sum_t y_{11}^2(t) & \sum_t y_{12}^2(t) \\ \sum_t y_{21}^2(t) & \sum_t y_{22}^2(t) \end{bmatrix} \text{ and it is } \begin{bmatrix} 0.0776 & 1.1534 \\ 1.0787 & 0.0946 \end{bmatrix} \text{ in our simulation.} \tag{22}$$

The *SNR* are 14.86 dB at receiver 1 and 11.40 dB at receiver 2 respectively, which is reasonable for the size of the filters employed. The two original signals, the two convolutively mixed signals and the recovered signals are all depicted in Figure 5. In listening tests, each output channel is dominated by a single recovered signal.

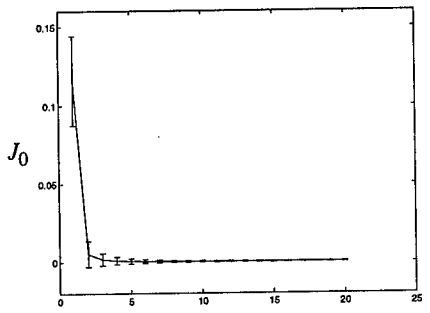


Figure 2. The performance of J_0 under minimization of J_0 versus epoch number

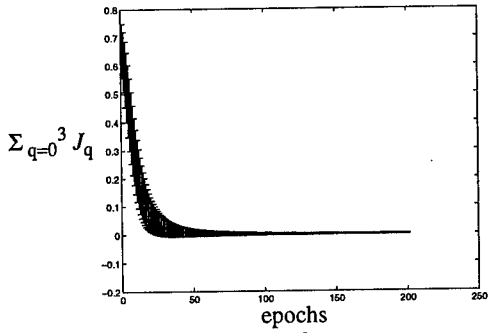


Figure 3. The performance of $\Sigma_{q=0}^{3} J_q$ under minimization of $\Sigma_{q=0}^{3} J_q$ versus epoch number

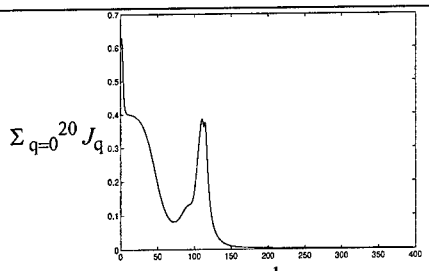


Figure 4. The performance of $\Sigma_{q=0}^{20} J_q$ under minimization of $\Sigma_{q=0}^{20} J_q$ versus epoch number

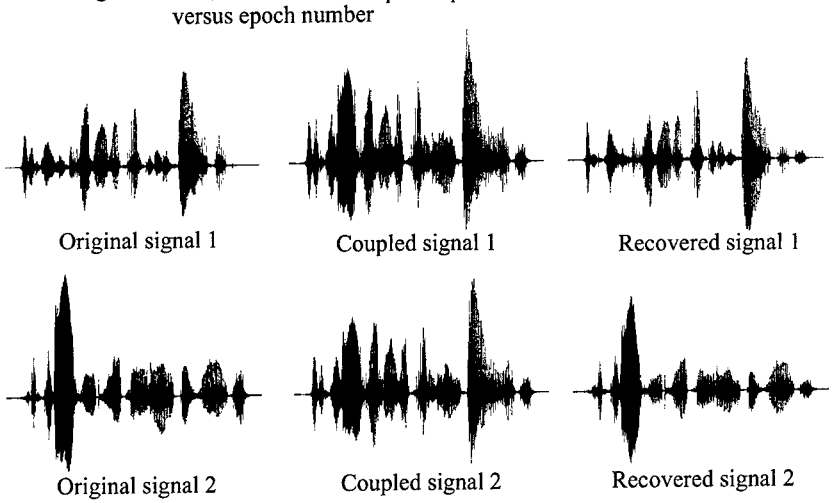


Figure 5. Signals for the simulation of linear convolutive mixture

6. Conclusion

We propose a generalized criterion based on the Frobenius norm for both blind source separation and decorrelation lifting a previous restriction on the positive definiteness of covariance matrices. Since the method is based on the minimization of a cost function it leads to on-line adaptation algorithms. However, the algorithm is not local and the computational complexity is $O(N^2)$, where N is the size of the network. The method displays fast and robust convergence as shown by Monte Carlo runs. The method is applicable to nonstationary signals like speech. If the signals are assumed stationary, the time-delayed decorrelation algorithm can be also used as an iterative alternative for blind separation similar to [7]. The cost function was extended to convolutive mixtures.

Acknowledgments: This work was partially supported by ONR N00014-94-1-0858, and NSF ECS-9510715

References

[1] E. Weinstein, M. Feder and A. V. Oppenheim, "Multi-channel signal separation by decorrelation," *IEEE Trans. on Speech and Audio Processing,* vol. 1, no. 4, pp. 405-413, October 1993.

[2] S. V. Gerven and D. Van Compernolle, "Signal separation by symmetric adaptive decorrelation: stability, convergence, and uniqueness," *IEEE Trans. on Signal Processing*, vol. 43, no. 7, pp. 1602-1611, July 1995.

[3] M. Najar, M. Lagunas, and I. Bonet, "Blind wideband source separation," in *Proc. 1994 IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Adelaide, South Australia, April 1994, pp. 65-68.

[4] A. Souloumiac, "Blind source detection and separation using second order nonstationary," in *Proc. 1995 IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Detroit, U. S. A., May 1995, pp. 1912-1915.

[5] T. Lo, H. Leung and J. Litva, "Separation of a mixture of chaotic signals," in *Proc. 1996 IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Atlanta, U. S. A., May 1996, pp. 1798-1801.

[6] K. Matsuoka, M. Ohya and M. Kawamoto, "A neural net for blind separation of nonstationary signals," *Neural Networks*, vol. 8, no. 3, pp. 411-419, 1995.

[7] L. Molgedey and H. G. Schuster, "Separation of a mixture of independent signals using time delayed correlations," Physical Review Letters, vol. 72, no. 23, pp. 3634-3637, June 1994.

[8] D. Chan, P. Rayner, S. J. Godsill, "Multi-channel signal separation," in *Proc. 1996 IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Atlanta, U. S. A., May 1996, pp. 649-652.

[9] C. Wang, H. Wu and J. Principe, "Correlation estimation using teacher forcing Hebbian learning and its application," in *Proc. 1996 IEEE Int. Conf. on Neural Networks*, Washington D. C., U. S. A., June 1996, pp. 282-287.

[10] A. Cichocki, W. Kasprzak and S. Amari, "Multi-layer neural networks with a local adaptive learning rule for blind separation of source signals," in *Proc. 1995 IEEE Int. Symp. on Nonlinear Theory App.*, Las Vegas, U. S. A., December 1995, pp. 61-65.

[11] S. Douglas and A. Cichocki, "Convergence analysis of local algorithms for blind decorrelation," *Technical report*, 1996.

[12] F. Silva and L. Almeida, "On the stability of symmetric adaptive decorrelation," in *Proc. 1994 IEEE Int. Conf. on Neural Networks*, Orlando, U. S. A., June 1994, pp. 66-71.

[13] L. Tong, R. Liu, V. Soon and Y. Huang, "Indeterminacy and identifiability of blind identification," *IEEE Trans. on Circuits and Systems*, vol. 38, no. 5, pp. 499-509, May 1991.

[14] G. Golub and C. Van Loan, *Matrix Computations*, second edition, John Hopkins Univ. Press, Baltimore, U. S. A., 1993.

[15] D. Yellin and E. Weinstein, "Criteria for multichannel signal separation," *IEEE Trans. on Signal Processing,* vol. 42, no. 8, pp. 2158-2168, August 1994.

APPLICATIONS

Invited Lecture

Neural Networks for Intelligent Multimedia Processing

S.Y. Kung

Princeton University

Multimedia technologies represent a new ground for research interactions among a variety of medias such as speech, audio, image, video, text, and graphics. Future multimedia technologies will need to handle information from multiple signal sources with an increasing level of intelligence, i.e. automatic recognition capabilities. On the other hand, neural processing is well-known to be an attractive means for implementing robust pattern recognition. It offers a good means for image and video segmentation and content-based indexing and retrieval. Unsupervised clustering and training by example are popular neural learning mechanisms. By these, machines may be taught to interpret possible variations of an object: e.g. scale, orientation/rotation, translation, contrast, and perspective. Ultimately, machines can be "trained" to see or hear, to recognize objects or faces, and to perceive human gestures or even emotions. In addition to adaptive learning, other useful characteristics include layered or hierarchical neural models and spatial/temporal processing models (as in temporal and static neural network structures). Some neural models have also effectively incorporated statistical signal processing (expectation-maximization, Gaussian mixtures) and optimization techniques (annealing). These key features of neural information processing have proven to be instrumental and effective to many applications in *intelligent multimedia processing (IMP)*. It is therefore envisioned that a major impact may be achieved by integrating adaptive neural processing into the state-of-the-art multimedia technologies. Neural processing and IMP share the following characteristics:

Digital Media Carrying Voluminous Spatial-Temporal Data: Tons of audiovisual information are available in digital form, accessible via internet to/from all places around the world.

Multi-Modality (Multiple Sensor/Data Sources): Joint processing of multimedia could result in significant advantages.

Trend Towards Intelligent Information Processing: So far, only text-based search engines are available on the WWW. (In fact, they are among the most frequently visited sites.) Some multimedia databases can offer very limited searching capabilities for pictures via color/texture features or information about the shape of objects in the picture. Advanced and more reliable indexing and retrieval techniques are not yet available on the market.

Many "laboratory" successes of neural networks for IMP applications have been quite encouragingly, reported. Examples include speech recognition/understanding; character recognition; texture classification; image/video segmentation; face-object detection/recognition; tracking of 3D objects; and lip reading via multi-modality combining visual and acoustic processing.

The potential applications of neural networks for IMP spread over a much broader spectrum: education (remote learning); shopping (searching clothes or fashion designs or a 3D house model); digital libraries (image catalog); journalism and multimedia editing (personalized electronic news service; media authoring; searching video clips of a celebrity) Multimedia directory services (yellow pages); entertainment and medical applications; etc. For example, it is not possible to efficiently search the web for, say, a sample picture or video clip, but shot from a different angle. In fact, there exists no generally recognized description of audiovisual contents. The research frontier today is gradually moving from what was primarily focused on coding (MPEG2 and MPEG4) to a new focus on automatic recognition. This trend is precipitated by a new member of the MPEG family: MPEG-7. MPEG-7 focuses on "multimedia content description interface". Its goal is to extend the current search capabilities to include more information types. Specifically, MPEG-7 will specify a standardized description of various types of multimedia information, including: still pictures, graphics, audio, moving video, and information about how these elements are combined in a multimedia presentation ('scenarios', composition information). This description shall be associated with the content itself, to facilitate fast and efficient searching for all the aforementioned medias. MPEG7 research domain will cover techniques for content-based indexing and retrieval: pattern recognition, face detection/recognition, fusion of multi-modality. For these, neural networks offer a very promising core technology.

Nonlinear Prediction of Chaotic Time Series Using Support Vector Machines

Sayan Mukherjee sayan@ai.mit.edu
Edgar Osuna eosuna@ai.mit.edu
Federico Girosi girosi@ai.mit.edu
Center for Biological and Computational Learning E25-201
Massachusetts Institute of Technology
Cambridge, MA 02139

Abstract

A novel method for regression has been recently proposed by V. Vapnik et al. [8, 9]. The technique, called Support Vector Machine (SVM), is very well founded from the mathematical point of view and seems to provide a new insight in function approximation. We implemented the SVM and tested it on the same data base of chaotic time series that was used in [1] to compare the performances of different approximation techniques, including polynomial and rational approximation, local polynomial techniques, Radial Basis Functions, and Neural Networks. The SVM performs better than the approaches presented in [1]. We also study, for a particular time series, the variability in performance with respect to the few free parameters of SVM.

1 Introduction

In this paper we analyze the performance of a new regression technique called a *Support Vector Machine* [8, 9]. This technique can be seen as a new way to train polynomial, neural network, or Radial Basis Functions regressors. The main difference between this technique and many conventional regression techniques is that it uses the *Structural Risk Minimization* and not the *Empirical Risk Minimization* induction principle. Since this is equivalent to minimizing an upper bound on the generalization error, rather than minimizing the training error, this technique is expected to perform better than conventional techniques. Our results show that SVM is a very promising regression technique, but in order to assess its reliability and performances more extensive experimentation will need to be done in the future. We begin by applying SVM to several chaotic time series data sets that were used by Casdagli [1] to test and compare the performances of different approximation techniques. The SVM is a technique with few free parameters. In absence of a principled way to choose these parameters we performed an experimental study to examine the variability in performance as some of these parameters vary between reasonable limits. The paper is organized as follows. In the next section we formulate the problem of time series prediction and see how it is equivalent to a regression problem. In section 3 we briefly review the SVM approach to the regression problem. In section 4, the chaotic time series

0-7803-4256-9/97/$10.00

used to benchmark previous regression methods [1] are introduced. Section 5 contains the experimental results and the comparison with the techniques presented in [1]. Section 6 focuses on a particular series, the Mackey-Glass, and examines the relation between parameters of the SVM and generalization error.

2 Time Series Prediction and Dynamical Systems

For the purpose of this paper a *dynamical system* is a smooth map $F : R \times \mathcal{S} \rightarrow \mathcal{S}$ where $\mathcal{S}$ is an open set of an Euclidean space. Writing $F(t, \mathbf{x}) = F_t(\mathbf{x})$, the map F has to satisfy the following conditions:

1. $F_0(\mathbf{x}) = \mathbf{x}$;
2. $F_t(F_s(\mathbf{x})) = F_{s+t}(\mathbf{x}) \quad \forall s, t \in R$

For any given initial condition $\mathbf{x}_0 = F_0(\mathbf{x})$ a dynamical system defines a trajectory $\mathbf{x}(t) = F_t(\mathbf{x}_0)$ in the set $\mathcal{S}$. The *direct* problem in dynamical systems consists in analyzing the behavior and the properties of the trajectories $\mathbf{x}(t)$ for different initial conditions $\mathbf{x}_0$. We are interested in a problem similar to the inverse of the problem stated above. We are given a finite portion of a time series $x(t)$, where x is a component of a vector $\mathbf{x}$ that represents a variable evolving according to some unknown dynamical system. We assume that the trajectory $\mathbf{x}(t)$ lies on a manifold with fractal dimension D (a "strange attractor"). Our goal is to be able to predict the future behavior of the time series $x(t)$. Remarkably, this can be done, at least in principle, without knowledge of the other components of the vector $\mathbf{x}(t)$. In fact, Takens embedding theorem [7] ensures that, under certain conditions, for almost all τ and for some $m \leq 2D+1$ there is a smooth map $f : \mathcal{R}^m \rightarrow \mathcal{R}$ such that:

$$x(n\tau) = f\left(x((n-1)\tau),\ x((n-2)\tau), \ldots, x((n-m)\tau)\right) \tag{1}$$

The value of m used is called the embedding dimension and the smallest value for which (1) is true is called the minimum embedding dimension, m^*. Therefore, if the map f were known, the value of x at time $n\tau$ is uniquely determined by its m values in the past. For simplicity of notation we define the m-dimensional vector

$$\tilde{\mathbf{x}}_{n-1} \equiv \left(x((n-1)\tau),\ x((n-2)\tau), \ldots, x((n-m)\tau)\right)$$

in such a way that eq. (1) can be written simply as $x(n\tau) = f(\tilde{\mathbf{x}}_{n-1})$. If N observations $\{x(n\tau)\}_{n=1}^{N}$ of the time series $x(t)$ are known, then one also knows $N - m$ values of the function f, and the problem of learning the dynamical system becomes equivalent to the problem of estimating the unknown function f from a set of $N - m$ sparse data points in $\mathcal{R}^m$. Many

regression techniques can be used to solve problems of this type. In this paper we concentrate on the Support Vector algorithm, a novel regression technique developed by V. Vapnik et al. [9].

3 Support Vectors Machines for Regression

In this section we sketch the ideas behind the Support Vectors Machines (SVM) for regression, a more detailed description can be found in [9] and [8]. In a regression problem we are given a data set $G = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, obtained sampling, with noise, some unknown function $g(\mathbf{x})$ and we are asked to determine a function f that approximates $g(\mathbf{x})$, based on the knowledge of G. The SVM considers approximating functions of the form:

$$f(\mathbf{x}, \mathbf{c}) = \sum_{i=1}^{D} c_i \phi_i(\mathbf{x}) + b \tag{2}$$

where the functions $\{\phi_i(\mathbf{x})\}_{i=1}^D$ are called *features*, and b and $\{c_i\}_{i=1}^\infty$ are coefficients that have to be estimated from the data. This form of approximation can be considered as an hyperplane in the D-dimensional feature space defined by the functions $\phi_i(\mathbf{x})$. The dimensionality of the feature space is not necessarily finite, and we will present examples in which it is infinite. The unknown coefficients are estimated by minimizing the following functional:

$$R(\mathbf{c}) = \frac{1}{N} \sum_{i=1}^{N} \mid y_i - f(\mathbf{x}_i, \mathbf{c}) \mid_\epsilon + \lambda \|\mathbf{c}\|^2 \tag{3}$$

where λ is a constant and the following *robust* error function has been defined:

$$\mid y_i - f(\mathbf{x}_i, \mathbf{c}) \mid_\epsilon = \begin{cases} 0 & \text{if } \mid y_i - f(\mathbf{x}_i, \mathbf{c}) \mid < \epsilon \\ \mid y_i - f(\mathbf{x}_i, \mathbf{c}) \mid & \text{otherwise.} \end{cases} \tag{4}$$

Vapnik showed in [8] that the function that minimizes the functional in eq. (3) depends on a finite number of parameters, and has the following form:

$$f(\mathbf{x}, \alpha, \alpha^*) = \sum_{i=1}^{N} (\alpha_i^* - \alpha_i) K(\mathbf{x}, \mathbf{x}_i) + b, \tag{5}$$

where $\alpha_i^* \alpha_i = 0$, $\alpha_i, \alpha_i^* \geq 0 \;\; i = 1, \ldots, N$, and $K(\mathbf{x}, \mathbf{y})$ is the so called *kernel* function, and describes the inner product in the D-dimensional feature space:

$$K(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{D} \phi_i(\mathbf{x}) \phi_i(\mathbf{y})$$

The interesting fact is that for many choices of the set $\{\phi_i(\mathbf{x})\}_{i=1}^D$, including infinite dimensional sets, the form of K is analytically known and very

simple, and the features ϕ_i never need to be computed in practice because the algorithm relies only on computation of scalar products in the feature space. Several choices for the kernel K are available, including gaussians, tensor product B-splines and trigonometric polynomials. The coefficients α and α^* are obtained by maximizing the following quadratic form:

$$R(\alpha^*, \alpha) = -\epsilon \sum_{i=1}^{N} (\alpha_i^* + \alpha_i) + \sum_{i=1}^{N} y_i (\alpha_i^* - \alpha_i) - \frac{1}{2} \sum_{i,j=1}^{N} (\alpha_i^* + \alpha_i)(\alpha_i^* - \alpha_i) K(\mathbf{x}_i, \mathbf{x}_j), \tag{6}$$

subject to the constraints $0 \leq \alpha_i^*, \alpha_i \leq C$ and $\sum_{i=1}^{N} (\alpha_i^* - \alpha_i) = 0$. Due to the nature of this quadratic programming problem, only a number of coefficients $\alpha_i^* - \alpha_i$ will be different from zero, and the data points associated to them are called *support vectors*. The parameters C and ϵ are two free parameters of the theory, and their choice is left to the user. They both control the VC-dimension of the approximation scheme, but in different ways. A clear theoretical understanding is still missing and we plan to conduct experimental work to understand their role.

4 Benchmark Time Series

We tested the SVM regression technique on the same set of chaotic time series that has been used in [1] to test and compare several approximation techniques.

4.1 The Mackey-Glass time series

We considered two time series generated by the Mackey-Glass delay-differential equation [4]:

$$\frac{dx(t)}{dt} = -0.1x(t) + \frac{0.2x(t-\Delta)}{1 + x(t-\Delta)^{10}}, \tag{7}$$

with parameters $\Delta = 17, 30$ and embedding dimensions $m = 4, 6$ respectively. We denote these two time-series by MG_{17} and MG_{30}. In order to be consistent with [1] the initial condition for the above equation was $x(t) = 0.9$ for $0 \leq t \leq \Delta$, and the sampling rate $\tau = 6$. The series were generated by numerical integration using a fourth order Runge-Kutta method.

4.2 The Ikeda map

The Ikeda map [2] is a two dimensional time series which is generated iterating the following map:

$$f(x_1, x_2) = (1 + \mu(x_1 \cos\omega - x_2 \sin\omega), \mu(x_1 \sin\omega + x_2 \cos\omega)), \tag{8}$$

where $\omega = 0.4 - 6.0/(1 + x_1^2 + x_2^2)$. In [1] Casdagli considered both this time series, that we will denote by $Ikeda_1$, and the one generated by the fourth iterate of this map, which has a more complicated dynamic, and that will be denoted by $Ikeda_4$.

4.3 The Lorenz time series

We also considered the time series associated to the variable x of the Lorenz differential equation [3]:

$$\dot{x} = \sigma(y - x), \quad \dot{y} = rx - y - xz, \quad \dot{z} = xy - bz \tag{9}$$

where $\sigma = 10$, $b = \frac{8}{3}$, and $r = 28$. We considered two different sampling rates, $\tau = 0.05$ and $\tau = 0.20$, generating the two time series $Lorenz_{0.05}$ and $Lorenz_{0.20}$. The series were generated by numerical integration using a fourth order Runge-Kutta method.

5 Comparison with Other Techniques

In this section we report the results of the SVM on the time series presented above, and compare them with the results reported in [1] about different approximation techniques (polynomial, rational, local polynomial, Radial Basis Functions with multiquadrics as basis function, and Neural Networks). In all cases a time series $\{x(n\tau)\}_{n=1}^{N+M}$, was generated: the first N points were used for training and the remaining M points were used for testing. In all cases N was set to 500, except for the $Ikeda_4$, for which $N = 100$, while M was always set to 1000. The data sets we used were the same that were used in [1]. Following [1], denoting by $\tilde{f}_N$ the predictor built using N data points, the following quantity was used as a measure of the generalization error of $\tilde{f}_N$:

$$\sigma^2(\tilde{f}_N) = \frac{1}{M} \sum_{n=N+1}^{M} \frac{(x(n\tau) - \tilde{f}_N(\tilde{\mathbf{x}}_{n-1}))^2}{\text{Var}} \tag{10}$$

where Var is the variance of the time series. We implemented the SVM using MINOS 5.4 [5] as the solver for the Quadratic Programming problem of eq. (6). Details of our implementation can be found in [6]. For each series we choose the kernel, K, and parameters of the kernel that gave us the smallest generalization error. This is consistent with the strategy adopted in [1]. The results are reported in table (1). The last column of the table contains the results of our experiments, while the rest of the table is from [1] with parameters and kernels set as in the remaining part of this section.

Mackey-Glass time-series: the kernel K was chosen to have the following form:

$$K(\mathbf{x}, \mathbf{x}_i) = \sum_{d=1}^{m} \frac{\sin((\nu + 1/2)(x^{(d)} - x_i^{(d)}))}{\sin(0.5(x^{(d)} - x_i^{(d)}))} \tag{11}$$

where $x^{(d)}$ is the d-th component of the m-dimensional vector $\mathbf{x}$, and a ν is an integer. This kernel generates an approximating function that is an additive trigonometric polynomial of degree ν, and correspond to features ϕ_i that are trigonometric monomials up to degree ν. We tried various values for ϵ and C. The embedding dimension for the series MG_{17} and MG_{30} were $m = 4$ and $m = 6$ in accordance with the work by Casdagli, and we used $\nu = 200$ and $\epsilon = 10^{-3}$.

Lorenz time-series: For the Lorenz$_0$.05 and Lorenz$_0$.20 series the polynomial kernels of order 6 and 10 were used. The embedding dimensions used were 6 and 10 respectively. The value of ϵ used was 10^{-3}.

Ikeda map: a B-spline of order 3 was used as the kernel, and the value of ϵ was 10^{-3}.

	Poly	Rational	Loc$^{d=1}$	Loc$^{d=2}$	RBF	N.Net	SVM
MG_{17}	-1.95$^{(7)}$	-1.14$^{(2)}$	-1.48	-1.89	-1.97	-2.00	-2.36 (258)
MG_{30}	-1.40$^{(4)}$	-1.33$^{(2)}$	-1.24	-1.42	-1.60	-1.5	-1.87 (341)
$Ikeda_1$	-5.57$^{(12)}$	-8.01$^{(8)}$	-1.71	-2.34	-2.95	-	-6.21 (374)
$Ikeda_4$	-1.05$^{(14)}$	-1.39$^{(14)}$	-1.26	-1.60	-2.10	-	-2.31 (427)
$Lor_{0.05}$	-4.62$^{(6)}$	-4.30$^{(3)}$	-2.00	-3.48	-3.54	-	-4.76 (389)
$Lor_{0.20}$	-1.05$^{(5)}$	-1.39$^{(6)}$	-1.26	-1.60	-2.10	-	-2.21 (448)

Table 1: Estimated values of $\log_{10} \sigma(\tilde{f}_n)$ for the SVM algorithm and for various regression algorithms, as reported in [1]. The degrees used for the best rational and polynomial regressors are in superscripts beside the estimates. Loc$^{d=1}$ and Loc$^{d=2}$ refer to local approximation with polynomials of degree 1 and 2 respectively. The numbers in parenthesis near the SVM estimates are the number of support vectors obtained by the algorithm. The Neural Networks results which are missing were also missing in [1].

6 Sensitivity of SVM to Parameters and Embedding Dimension

In this section we report our observations on how the generalization error and the number of support vectors vary with respect to the free parameters of the SVM and to the choice of the embedding dimension. The parameters

we analyze are therefore C, ϵ, the dimensionality of the feature space D, and the embedding dimension m. All of these results are for the MG_{17} series. Figure 1a demonstrates that C has little effect on the generalization error (the plot spans over 7 orders of magnitude). The parameter C has also little effect on the number of support vector, as shown in figure 1b, which remains almost constant in the range $10^{-2} - 10^{2}$. The results were similar for kernels with low ($D = 2$), high ($D = 802$) and infinite dimensionality of the feature spaces.

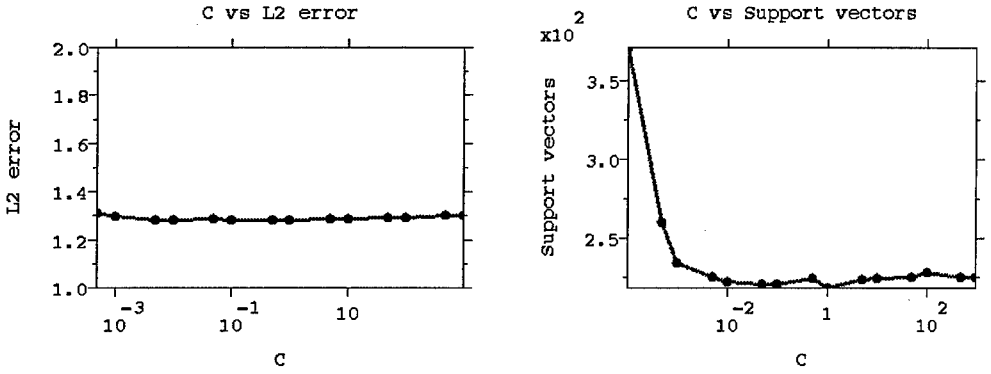


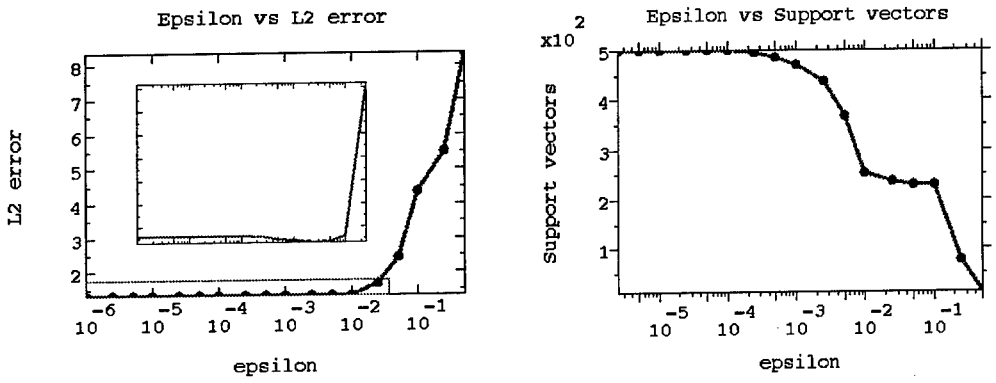
Figure 1: (a) The L^2 generalization error versus C for the MG_{17} series. (b) The number of support vectors versus C for the same series. The kernel was an additive trigonometric polynomial with $\nu = 200$.

The parameter ϵ has a strong effect on the number of support vectors and on the generalization error, and its relevance is related to D. In order to see why this happens, remember that if $I[\mathbf{c}]$ and $I_{\text{emp}}[\mathbf{c}]$ are respectively the expected risk and empirical risk, with probability $1 - \eta$:

$$I[\mathbf{c}] \leq I_{\text{emp}}[\mathbf{c}] + \tau \sqrt{\frac{h(\log \frac{2N}{h} + 1) - \log \frac{\eta}{4}}{N}}, \tag{12}$$

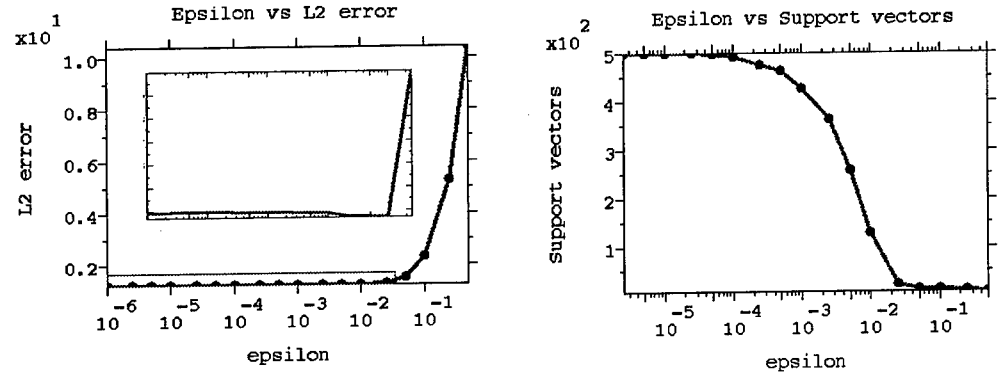
where τ is a bound on the cost function used to define the expected risk and h is the VC-dimension of the approximation scheme. It is known that the VC-dimension satisfies $h \leq \min(\frac{R^2(A+1)^2}{\epsilon^2}, D) + 1$[10], where R is the radius of the smallest sphere that contains all the data points in the feature space, A is a bound on the norm of the vector of coefficients. When D is small, the VC-dimension h is not dependent on ϵ and the second term on the bound of the generalization error is constant and therefore a very small ϵ does not cause overfitting. For the same reason when D is large the term is very sensitive to ϵ and overfitting occurs for small ϵ. Numerical results confirm this. For example, figures 2 and 3 which correspond to feature spaces of 802

and infinite dimensions, respectively, show overfitting. (The kernels used were the additive trigonometric polynomial with $\nu = 200$ and a B-spline of order 3, respectively.) Figure 4 corresponds to a feature space of 10 dimensions and there is no overfitting. (The kernel used was the additive trigonometric polynomial with $\nu = 2$.)



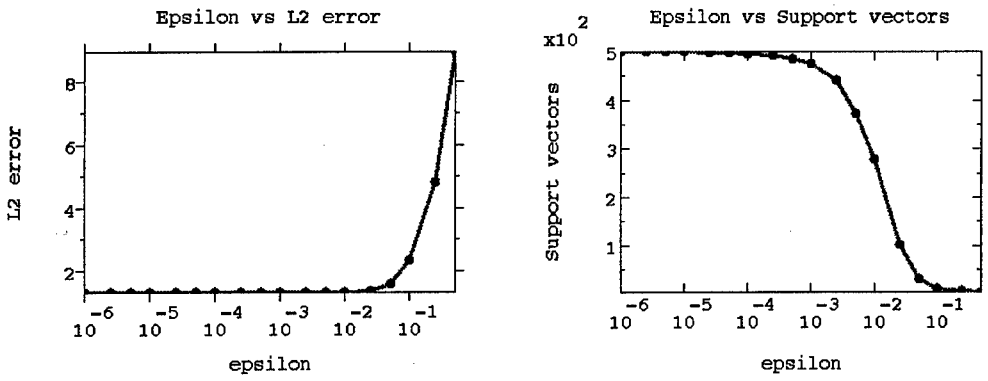
(a) (b)

Figure 2: (a) The L^2 generalization error versus ϵ with a 802 dimensional feature space. The inset magnifies the boxed region in the lower left section of the plot. Note that overfitting occurs. (b) The number of support vectors versus ϵ for the same feature space.



(a) (b)

Figure 3: (a) The L^2 generalization error versus ϵ with an infinite dimensional feature space. The inset magnifies the boxed region in the lower left section of the plot. Note that ovefitting occurs (b) The number of support vectors versus ϵ for the same feature space.



(a) (b)

Figure 4: (a) The L^2 generalization error versus ϵ with a 10 dimensional feature space. Note that there is no overfitting (b) The number of support vectors versus ϵ for the same feature space.

The effect of the embedding dimension m on generalization error was also examined. According to Takens theorem the generalization error should decrease as m approaches the minimum embedding dimension, m^*. Above m^* there should be no decrease in the generalization error. However, if the regression algorithm is sensitive to overfitting the generalization error can increase for $m > m^*$. The minimal embedding dimension of the MG_{17} series is 4. Our numerical results demonstrate the SVM does not overfit for the case of a low dimensional kernel and overfits slightly for high dimensional kernels, see figure 5. The additive trigonometric polynomial with $\nu = 200$ and 2 were used for this figure.

7 Discussion

The SVM algorithm showed excellent performances on the data base of chaotic time series, outperforming the other techniques in the benchmark in all but one case. The generalization error is not sensitive to the choice of C, and very stable with respect to ϵ in a wide range. The variability of the performances with ϵ and D seems consistent with the theory of VC bounds.

Acknowledgements

Figures in this paper were displayed with the Signiscope^R.

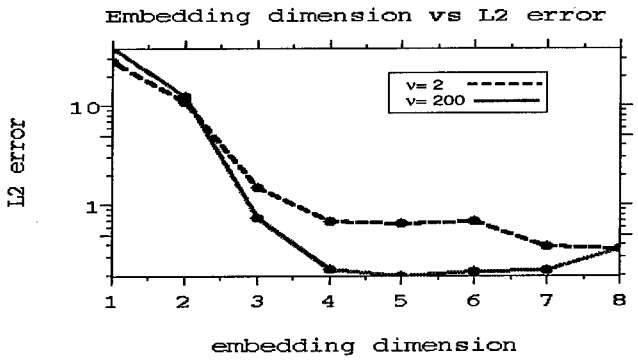


Figure 5: The L^2 generalization error versus the embedding dimension. The solid line is for a 802 dimensional feature space and the dashed line is for a 10 dimensional feature space.

References

[1] M. Casdagli. Nonlinear prediction of chaotic time-series. *Physica D*, 35:335–356, 1989.

[2] K. Ikeda. Multiple-valued stationary state and its instability of light by a ring cavity system. *Opt. Commun.*, 30:257, 1979.

[3] E.N. Lorenz. Deterministic non-periodic flow. *J. Atoms. Sci.*, 26:636, 1969.

[4] M.C. Mackey and J. Glass. Oscillation and chaos in physiological contol systems. *Science*, 197:287, 1977.

[5] B.A. Murtagh and M. Saunders. Large-scale linearly constrained optimization. *Mathematical Programming*, 14:41–72, 1978.

[6] E. Osuna, R. Freund, and F. Girosi. Support vector machines: Training and applications. A.I. Memo 1602, MIT A. I. Lab., 1997.

[7] F. Takens. Detecting strange attractors in fluid turbulence. In D. Rand and L.S. Young, editors, *Dynamical Systems and Turbulence*. Springer-Verlag, Berlin, 1981.

[8] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.

[9] V. Vapnik, S.E. Golowich, and A. Smola. Support vector method for function approximation, regression estimation, and signal processing. In *Advances in Neural information processings systems 8*, San Mateo, CA, 1996. Morgan Kaufmann Publishers. (in press).

[10] V. N. Vapnik. *Statistical learning theory*. J. Wiley, 1997.

A DCT-based Adaptive Metric Learning Model Using Asymptotic Local Information Measure

Takami Satonaka, Takaaki Baba, Takayuki Chikamura, Tatsuo Otsuki and Teresa H. Meng†
Stanford University† and Matsushita Electronics Co.
Gates R337, Stanford CA94305-9035
E-mail {satonaka@video,thm@kailas}.stanford.edu

Abstract

We present an adaptive metric learning vector quantization procedure based on the discrete-cosine transform (DCT) for accurate face recognition used in multimedia applications. Since the set of learning samples may be small, we employ a mixture model of prior distributions. The model selection method, which minimizes the cross entropy between the real distribution and the modeled one, is presented to optimize the mixture number and local metric parameters. The structural risk minimization is used to facilitate an asymptotic approximation of the cross entropy for models of fixed complexity. We also provide a formula to estimate the model complexity derived from the minimum description length criterion. The structural risk minimization method proposed achieves an recognition error rate of 2.29% using the ORL database, which is better than previously reported numbers using the Karhunen-Loeve transform convolution network, the hidden Marcov model and the eigenface model.

1. INTRODUCTION

In this paper, we describe an adaptive metric learning vector quantization (LVQ) using the discrete cosine transform (DCT) for face classification in multimedia applications, such as used in camera and facial signature recognition.

In the past, the Karhunen-Loeve(KL) transform and principal component analysis (PCA) have been successfully used for face feature detection [1][2]. We employ the DCT transform with the added advantage of having a computationally-efficient and data-independent matrix [3] as an alternative to the KL transform which requires data-dependence eigenvectors as a priori information.

Another approach of the LVQ procedure as described in [4] is an effective clustering method for a large set of training samples. However, the performance is degraded with learning from a small set of samples. A number of

0-7803-4256-9/97/$10.00 ©1997 IEEE

promising approaches considering local probability distribution [5][6][7][8] [9]have also been proposed. For example, Hastie presents the normal mixture model, assuming a common variance matrix for all classes. However, for the learning from an insufficient set of learning samples, the assumption of local distribution is crucial [9], which makes the choice on the number of mixture components extremely difficult [5].

For the learning from a small set of samples, we propose an adaptive metric LVQ, based on a mixture model of local prior distributions. Our model assigns a variable number of the mixture classes to each class and then determines the mixture number and local metric parameters to minimize the cross entropy between the real distribution and the modeled one. In the model selection, the minimum description length (MDL) [10] and the structural risk minimization (SRM) principles [11] are indispensable. The structural risk minimization of negative log-likelihood has been introduced as an asymptotic approximation to the cross entropy minimization in cases where the model parameters have fixed complexity. Moreover, we can estimate the model complexity kc and the data number *Np* using the complexity formula *kc(logNp) /Np* derived from the MDL criterion. It provides a good measure to make the trade-offs between accuracy and complexity in determining the length of DCT coefficients for the case of larger *kc/Np* ratio.

The local minimization of the entropy distance is demonstrated for face recognition of the ORL face database, which consists of 40 persons of 10 different poses with distinct variations such as open/close eyes, smiling/non-smiling faces, glasses/non-glasses poses, and rotation up to 20 degrees. The results are compared with those obtained from using the hidden Marcov model [12], PCA, convolution network [13] based on KLT features, and Kohonen's self-organization map.

2. ADAPTIVE METRIC MODEL

2.1. The Learning Algorithm

An adaptive metric LVQ estimates the Mahalanobis distance, $D(\bar{x}^P, \bar{x}_\mu^k)$, between an input vector $\bar{x}^P = (x^{p,1}, \cdots, x^{p,Np})$ and a reference vector $\bar{x}_\mu^k = (x_\mu^{k,1}, \cdots, x_\mu^{k,Np})$, assuming

$$P(\theta^k) = \frac{1}{\sqrt{2\pi\Sigma_k}} \exp(\frac{1}{2} D(\bar{x}^P, \bar{x}_\mu^k)),$$
$$= \frac{1}{\sqrt{2\pi\Sigma_k}} \exp(\frac{1}{2} (\bar{x}^P - \bar{x}_\mu^k)^T \Sigma_k^{-1} (\bar{x}^P - \bar{x}_\mu^k)). \quad (1)$$

Each component of the input vector is assumed to be independently and identically distributed (i.i.d). Therefore, $|\Sigma_k|_{(i,j)} = \delta_{(i,j)}\sigma^{k,j}$ and $P(\theta^k) = \prod_{i=1}^{Np} P(\theta^{k,i})$.

For each presentation of an input vector in class p, we estimate the Mahalanobis distance and update components of reference vectors $\bar{x}_\mu^k$ and $\bar{x}_\mu^l$ for the first and second nearest neighbor classes k and l, respectively, as depicted in the following equations:

$$x_{\mu(t+1)}^{k,i} = x_{\mu(t)}^{k,i} + \beta(x^{p,i} - x_{\mu(t)}^{k,i}) \quad \text{for } p=k. \tag{2}$$

$$x_{\mu(t+1)}^{l,i} = x_{\mu(t)}^{l,i} - \beta(x^{p,i} - x_{\mu(t)}^{l,i}) \quad \text{for } p\neq l. \tag{3}$$

2.2. Local Information Measure

To obtain optimal variances from small learning sets, we propose a mixture model using the minimization of the cross entropy between an unknown true distribution $P(x^{k,i})$ and a modeled one $Q(x^{k,i})$, which is equal to $-logP(x^{k,i})$. We deal with the empirical risk function $\sum_{p=1}^{Np} Q_q(x_p^{k,i})/N_p$ in the place of the cross entropy $\int P(x^{k,i})\, Q(x^{k,i})\, dx^{k,i}$ for an i.i.d sample sequence of $x^{k,i}{}_1, \ldots x^{k,i}{}_{Np}$. According to the Vapnik-Chervonenkis theory, the error is bounded by

$$P\left\{\sup\left|\int P(x^{k,i})Q(x^{k,i})dx - \frac{1}{Np}\sum_{p=1}^{Np} Q_q(x_p^{k,i})\right| > \varepsilon\right\} \le 4\exp\left\{-\left(\varepsilon^2 + \frac{h\left(\ln(2N_p/h)+1\right)}{N_p}\right)N_p\right\}. \tag{4}$$

The h is the Vapnik-Chervonenkis dimension denoting parametric complexity and N_p is the sample size. We obtain asymptotically optimal variances from the mixture distributions by increasing the size of training samples even in cases where the learning parameters have a fixed complexity h.

Figure 1 shows a schematic illustration of structure of mixture classes. Our model assumes the $(q+1)$-fold product of mixture distribution, $P_q(\sigma_q^{k,i}, x^{k,i}) = \Pi^{q}_{j=0} P(\sigma^{m(j),i}, x^{m(j),i})$, to derive the asymptotically optimal variance $\sigma_q^{k,i}$ of component i in class k. Let the $(q+1)$-fold mixture classes $M^{k,i}{}_q$ be provided with a structure:

$$M_1^{k,i} \in M_2^{k,i} \cdots \in M_q^{k,i}, \qquad l_{\min(1)}^{k,i} \ge l_{\min(2)}^{k,i} \cdots \ge l_{\min(q)}^{k,i}. \tag{5}$$

$$\begin{aligned} l_{\min(q)}^{k,i} &= \frac{1}{(q+1)Np}\sum_{j=0}^{q}\sum_{p=1}^{Np} Q_q(\sigma^{m(j),i}, x_p^{m(j),i}), \\ &= -\frac{1}{(q+1)Np}\sum_{j=0}^{q}\sum_{p=1}^{Np} \log P(\sigma^{m(j),i}, x_p^{m(j),i}). \end{aligned} \tag{6}$$

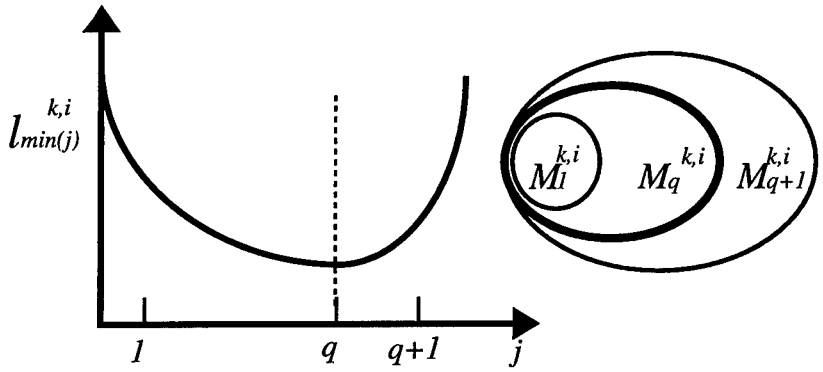


Figure 1. A schematic structure of mixture classes defined by using a likelihood function $l^{k,i}_{min(j)}$ of mixture number j.

The j-th mixture class $m(j)$ is assigned according to each reference class k based on the likelihood measure $l^{k,i}_{min(j)}$. For each class, the minimal size of the mixture model is uniquely derived from equation (5). Then, an optimal variance is obtained by

$$\sigma_q^{k,i} = \frac{\sum_{j=0}^{j=q} \sigma^{m(j),i}}{q+1}. \tag{7}$$

Furthermore we can extend this model by introducing the complexity term $(k_c \log N_p / 2 + \log M) / N_p$ based on the minimum description length(MDL) criterion:

$$L^{k,i}_{\min(q)} = \frac{1}{N_p}\{ l^{k,i}_{\min(q)} + \frac{k_c}{2}\log N_p + \log M \}. \tag{8}$$

The k_c and M denote the dimension of the model and the maximum number of the model respectively. This is equivalent to the MDL criterion in providing a trade-off between correctness and complexity in selecting the distribution model .

2.3. Number of DCT Coefficients

The optimal number of coefficients in the DCT approximation of gray images can be derived from the MDL criterion:

$$l_{dct} = -\log(P(\sigma)) + \frac{k}{2}\log(N_p) + \log M, \tag{9}$$

assuming

$$P(\sigma)=\frac{k}{\sqrt{2\pi}\sigma}\exp(-\frac{1}{2\sigma^2}\sum_{x=1}^{N}\sum_{y=1}^{N}\varepsilon_{(x,y)}{}^2). \quad (10)$$

$$\varepsilon_{(x,y)}=f_{(x,y)}-\frac{2}{N}\sum_{u=1}^{Ndct}\sum_{u=1}^{Ndct}C_uC_vF_{(u,v)}\cos\frac{\pi(2x+1)u}{2N}\cos\frac{\pi(2y+1)v}{2N}. \quad (11)$$

The k, N_p and M denote the dimension of the model, the number of data samples and the maximum number of model classes. The $F_{(u,v)}$ is the discrete cosine transform of $f_{(x,y)}$ and the N_{dct} is number of the coefficients. The maximum likelihood DCT coefficients are obtained by minimizing the l_{dct} of equation (9).

3. RESULTS

The ORL database consists of 40 persons with 10 distinct facial expressions. The original image of 114x 88 pixels is transformed into an image of 16x16 pixels. The coefficients derived from the 16x16 DCT are used for learning.

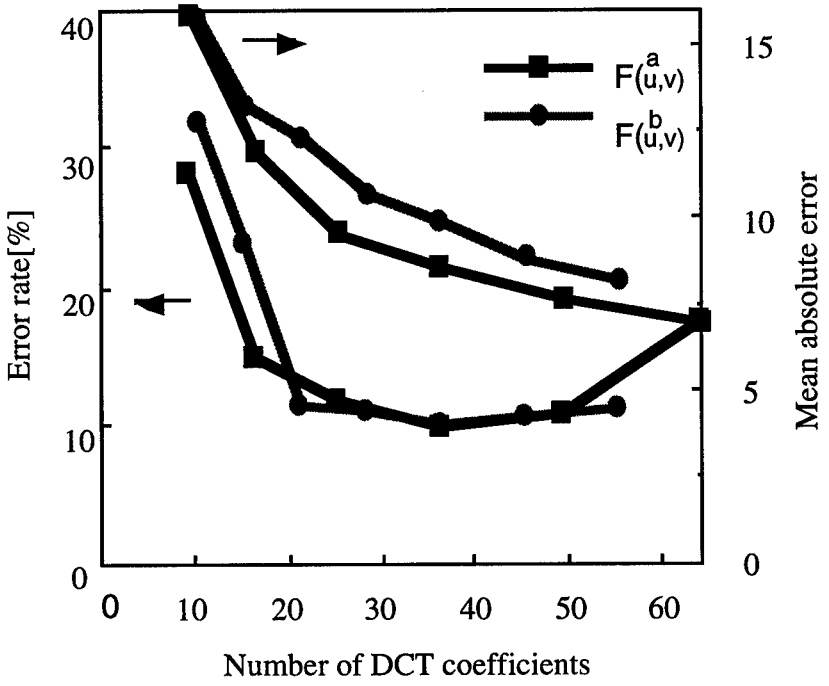


Figure 2. Relationships between the number of DCT coefficients and recognition error rate and mean absolute error.

First, we derive an optimal number of DCT coefficients. Figure 2 shows the relationships between the numbers of coefficients and the recognition error rate and mean absolute error for the coefficients $\{F^a_{(u,v)} \mid 0< u, v < N_{dct}\}$ and $\{F^b_{(u,v)} \mid 0 < u + v < N_{dct}\}$. It demonstrates the trade-off between error rate and complexity in the DCT approximation. Increasing the coefficient number reduces the approximation error but increases the complexity of the learning network by $k_c(logNp)$. The recognition error rates reach the minimum value at around 36 coefficients.

The main issue of constructing our learning model is to assume proper input pattern distributions. For examples, Hastie proposes a normal distribution sharing common component variance among all classes [5]. Table 1 shows error rate and description length obtained from the distributions with three kinds of variances. The distribution with a common component variance specified by (c2) [5] is compared with a distribution with different component variances specified by (c3) and with that of Kohonen LVQ with $\sigma^{k,i}=1$ (c1). The result of the distribution (c3) is worse than that of Kohonen LVQ with $\sigma^{k,i}=1$. The error rate obtained from the distribution (c2) is better than those obtained from (c1) and (c3). The learning performance significantly depends on the distribution assumptions. We ensure the goodness of the assumptions in terms of description length. The description length derived form the distributions (c2) and (c3) are 1084 and 1616, respectively. The assumption used in Hastie model has attained smaller description length by effectively reducing the complexity term in equation (8). The complexity terms derived from distributions (c2) and (c3) are $(k_c/2)\ \log N_p + \log(k_c N_p)$ and $k_c \log N_p$. (In our simulation, the number of classes k_c and the number N_p of patterns are 40 and 5, respectively.) The distribution proposed by Hastie [5] is more probable when the complexity term is reduced, although it is the very restrictive assumption.

Table 1. Error rate and description length obtained from the distributions with three kinds of variances.

	c1	c2	c3
Variance	$\sigma^{k,i}=1$	$\frac{\Sigma\sigma^{k,i}}{Np}$	$\sigma^{k,i}$
Error rate[%]	12.10	9.12	14.30
Description length		1084	1616

To reduce the first term of equation (8), our approach employs a mixture model of normal distributions allowing a variable number of mixture classes. To determine an optimal size, we explore minimization of several metric measures consisting of (m1) cross entropy, (m2) variance distance and (m3) mean value distance, which are defined by using normal distributions of classes k and l. Table2 shows the error rates and average variances obtained from fourfold mixture distributions. The asymptotic minimization of cross entropy is estimated by using $\Sigma(\sigma^{k,i2}+\sigma^{l,i2})$. The error rate obtained from the cross entropy minimization is 7.68%, which is better than 8.24% and 12.25% obtained from mean value distance and variance distance minimization.

Table2 Error rates and average variances obtained from fourfold mixture distributions.

	m1	m2	m3
Metric measure	$\Sigma(\sigma^{k,i2}+\sigma^{l,i2})$	$\Sigma \lvert\sigma^{k,i}-\sigma^{l,i}\rvert^2$	$\Sigma \lvert\mu^{k,i}-\mu^{l,i}\rvert^2$
Error rate[%]	7.68	12.25	8.24
Variance $\Sigma\sigma^{k,i}$	36.20	38.82	38.87

For component inputs of DCT coefficients, the entropy distance is defined by assuming a variable number of i.d.d mixture classes. Figure 3 shows the relation of average variance and error rate to the number of mixture classes. We compare the performance of using our model specified by (m4), with results derived from a fixed size of mixture classes for all the components specified by (m1) and (m2). The minimal error rates of 7.65% and 8.24% or the variances of 618 and 960 are obtained at the size of 3 and 4 for the entropy distance and the centroid distance. Using our model, the error rate and average variance are decreased and saturated to 6.28% and 516 with an increase in the number of i.i.d mixture classes. The essence of the structural risk minimization allows variable size of the mixture classes according to equation (5) for i.i.d DCT components. The error rate with variable size of mixture classes is significantly improved from 7.78% to 6.28% by allowing an optimal size of mixture classes defined by the entropy distance.

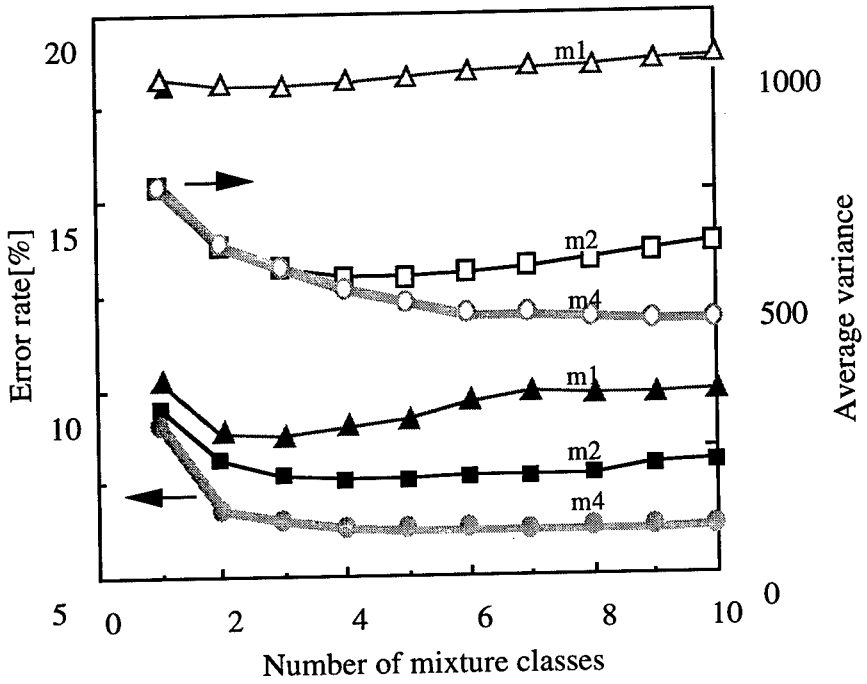


Figure 3. Relation of average variance and error rate to number of mixture classes.

Table 3. Face recognition results using 5 faces for training and 5 testing from the ORL database.

	DCT+ AMLVQ	SOM+ CN	KLT+ CN	Pseudo2D-HMM	Eigenfaces
Error rate[%]	2.29	3.8	5.3	5.0	10.5

Table 3 shows the face recognition results using 5 faces for training and 5 testing from the ORL database. Figure 4 shows the recognition error rates as a function of number of training faces. An error rate of 2.29% using our method is achieved, which is better than the 3.8%, 5.3%, 5% and 10.5% obtained using SOM+CN, KLT convolution network [13], Pseudo 2D hidden Markov model [12] and eigenface model [13], respectively. Furthermore the previous results were obtained with averaging two best trials, while our results were obtained with averaging the best five.

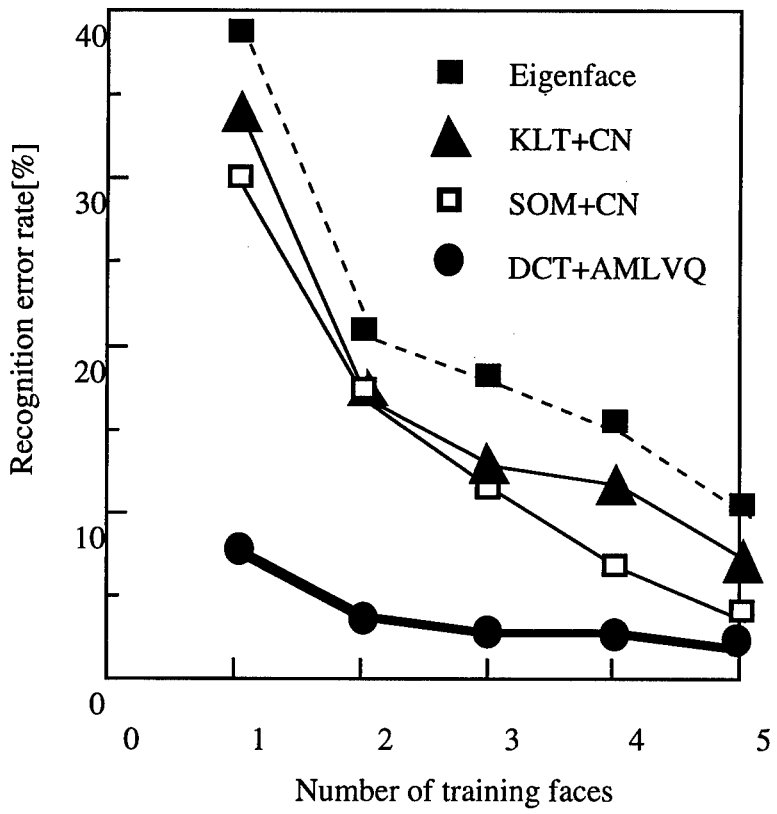


Figure 4. Recognition error rates as a function of number of training faces. The other faces of 10 faces are used for testing.

4. CONCLUSIONS

We present a DCT-based adaptive metric LVQ employing a mixture model designed for learning from small sample sets such as in face identification. Adaptive sizes of the mixture classes and local metric parameters are derived from the structural empirical risk minimization for a certain model complexity. We extend this model to provide a general trade-off between error and complexity by introducing the complexity term in MDL. The optimal size of the DCT coefficients is also obtained, achieving the lowest recognition error rate using the ORL database reported so far.

Acknowledgments

The authors would like to thank Managing Director Dr. Kano for warm support and the Olivetti Research Laboratory for maintaining the ORL database.

References

[1] S. Akamatsu, T. Sasaki, H. Fukamachi, and Y. Suenaga, "A robust face identification scheme-KL expansion of an invariant feature space." *SPIE Proc.:Intelligence. Robots and Computer Vision X: Algorithms and Techn.*, 1607:71-84,1991.

[2] A. Pentland and B. Moghaddam,T. Starner, and M. Turk, "View-based and modular eigenspaces for face recognition," *Proc. of the IEEE Computer Soc. Conf. on Computer Vision and Pattern Recognition*, pp 84-91, 1994.

[3] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*, Dordrecgt, Neherlands,1991,Kluwer Academic Publishers

[4] T. Kohonen, "The self-organizing map," *Proc. of the IEEE 78*, 1464-1480,1990.

[5] T. Hastie and R. Tibshirani, "Discriminate Analysis by Gaussian Mixture," *Journal of the Royal Statistical Society*, Vol.58, pp155-176, 1996.

[6] S.Y. Kung and J.S.Taur, "Decision-Based Neural Network with Signal/Image Classification Applications," *IEEE Trans. on Neural Networks*,6(1):170-181,1995

[7] T. Poggio and D. Beymer, "Regularization Networks for Visual leaning", in S. K. Nayar and T. Poggio editors, *Early Visual Learning*, New York, 1996. Oxford University Press, Inc.

[8] A.P. Dempster, N.M. Larid, D.B. Rubin, D.B., "Maximum Likelihood from Incomplete Data via the EM Algorithm," *Journal of the Royal Statistical Society B*, vol.39,pp.1-38,1977

[9] G. J. McLachlan and T. Krishnan, *The EM Algorithm and Extension*, NY, John Wiley & Sons, 1996

[10] J. Rissanen , "Universal Coding, Information, Prediction, and Estimation," *IEEE Trans. Inform. Theory*, vol. IT-30,NO.4, July 1984.

[11] V. N. Vapnik, *The Nature of Statistical Learning Theory , Information, Prediction, and Estimation*, NY, Springer-Verlag,1996.

[12] F. Samaria and A. Harter, "Parameterization of a stochastic model for human face Identification", *Proc. of the 2nd IEEE workshop on Application of Computer Vision*, Sarasota, Florida,1994.

[13] S. Lawrence, C. L. Giles, A. C. Tsoi and A. Back, "Face Recognition: A Hybrid Neural Network Approach," *Technical Report CS-TR-3608*, University of Maryland,1996.

SUB-WORD SPEAKER VERIFICATION USING DATA FUSION METHODS

Kevin R. Farrell, Ravi P. Ramachandran, Manish Sharma, and Richard J. Mammone

T-NETIX/SpeakEZ Inc.
67 Inverness Drive East
Englewood, Colorado 80112
Tel: 303-705-5556, FAX: 303-790-9540
email: kevin.farrell@denver.t-netix.com

ABSTRACT

Speaker verification is a rapidly maturing technology that is becoming available for commercial applications. In this paper, we investigate the application of data fusion methods to sub-word implementations of speaker verification. At a sub-word level, we utilize the diversity of the information provided by the neural tree network and Gaussian mixture model to provide a more robust sub-word model. The phrase-level scores for each modeling approach are obtained and then combined. The data fusion method we use for combining the model scores is the linear opinion pool. In addition to using the diversity of the model scores, we also apply the concept of redundancy by using a leave-one-out approach to partition the input data. This allows us to generate several models and accommodate the small training sample issues imposed by our specific applications. The theoretical results of the above analysis have been integrated into a system that has been tested with several databases that were collected within landline and cellular environments. These results are included in this paper. We have found that the proper data fusion techniques will typically reduce the error rate by a factor of two.

0-7803-4256-9/97/$10.00

1 INTRODUCTION

Speaker verification consists of determining whether or not a voice sample provides sufficient match to a claimed identity. Speaker verification has numerous applications in areas that necessitate the validation of a person's identity. For example, when initiating a bank account transaction over the phone or at an automatic teller machine (ATM), speaker verification can provide an additional level of security over personal identification numbers (PINs). Also, speaker verification has the advantage over other forms of biometric authentification, such as fingerprint, retinal scan, etc., in that it can be applied over the telephone network. These are some of the characteristics that make speaker verification a very attractive technology for numerous commercial applications.

Speaker verification applications are generally text-independent or text-dependent. Text-independent speaker verification systems do not require that the same text be used for training and testing. Text-dependent speaker verification systems require that the same text be used during both training and testing. Though text-independent systems may be more convenient from a user standpoint, text-dependent systems provide additional security in that they 1) require fraudulent imposter attempts to use the same password, and 2) tend to provide better performance than text-independent systems. Text-dependent speaker verification systems will be the focus of this paper.

In this paper, we investigate the application of data fusion methods to sub-word model implementations of text-dependent speaker verification. The effects of segmentation for sub-word implementations are addressed. Two modeling approaches are then considered for score combination, namely the neural tree network and Gaussian mixture model.

This paper is organized as follows. The following section provides an overview of the processing steps in performing speaker verification. This overview includes a brief description of feature extraction, model evaluation, and data fusion. This is followed by a description of the implementation details that are specific to our system. The experimental results for several text-dependent tasks are then provided. The databases used for these experiments are collected within both landline and cellular environments. A summary of the results is then given.

2 SPEAKER VERIFICATION

Speaker verification generally consists of feature extraction followed by model construction and evaluation. As part of model construction and evaluation, we will also address the concept of data fusion where the scores of several models are combined to create a composite score. This composite score will be that which is applied to a threshold to yield the final decision. These phases of speaker verification are briefly described in the following subsections.

2.1 Feature Extraction

Feature extraction consists of deriving characteristics of the speech signal that are unique to an individual. The predominant characteristic that causes people's voices to be different from one another is the shape of the vocal tract. The difference in the length and cross-sectional areas in the vocal tract from person to person results in different resonant frequencies and bandwidths. Hence, most feature extraction routines for speaker recognition utilize some type of spectral analysis. Typical features are the cepstrum or variants of it. Pole-filtered, mean-removed cepstrum [1] are the features used in the experimental results section. For this feature set we first obtain a channel estimate by computing the pole-filtered mean of the linear predictive (LP) cepstrum of the input speech. This channel estimate is converted to a filter that is applied to the speech to inverse out the channel effect. Then, the LP cepstrum of the filtered speech is used as the feature.

2.2 Modeling

A speaker verification model is constructed from feature data, specifically that from a target speaker and possibly from non-target speakers. This model should have the ability to provide a level of match to the target speaker when given a new set of feature data. For text-dependent speaker verification, a model should capture the temporal information in addition to the acoustical information. The standard models that accomplish this are hidden Markov models (HMMs) and dynamic time warping (DTW). In general, segment-based approaches to speaker verification maintain temporal information. Another important piece of information for model construction or evaluation is data that is not from the target speaker, or "non-target" data. One method for incorporating this information is used during model evaluation and is known as cohort normalization [2]. Another method is to use non-target data during training, which can be accomplished by using discriminative training approaches [3] or neural networks [4].

The modeling approach here is based on the neural tree network (NTN) and Gaussian mixture model (GMM). The NTN [5] is a hierarchical classifier that uses a tree architecture to implement a sequential linear decision strategy. The NTN has been evaluated for text-independent speaker verification [4], whole-word based, text-dependent speaker verification [6], and sub-word based, text-dependent speaker verification [7, 8]. Data fusion methods were considered for whole-word NTN models with dynamic time warping [6, 9]. In this paper, we evaluate data fusion methods for sub-word NTN models combined with Gaussian mixture modeling, which is also a popular model for speaker verification [10].

2.3 Data Fusion

Data fusion methods can take advantage of the concepts of diversity and redundancy to improve system performance. Diversity can be used to improve system performance through the incorporation of different information. Similarly, redundancy can achieve the same goals through the re-use of data. These concepts have been thoroughly explored in the field of communications and have also been applied to pattern recognition problems. The basic idea is that if several models can be constructed, whose errors are mutually uncorrelated, then performance advantages can be obtained through the proper combination of the model scores.

The combination of different sources of information has been explored within a field known as data fusion. A comparison was done between several data fusion techniques, including the linear and log opinion pools [11], and voting [12] for a speaker verification application [6]. This comparison showed the simplest method, namely the linear opinion pool, to do at least as well as the other methods. Hence, the linear opinion pool will be considered here. The linear opinion pool is evaluated as a weighted sum of the outputs for each model:

$$P_{linear}(x) = \sum_{i=1}^{n} \alpha_i p_i(x), \tag{1}$$

where $P_{linear}(x)$ is the probability of the combined system, α_i are weights, $p_i(x)$ is the probability output by the i^{th} model, and n is the number of models. For all experiments in this paper, α_i is between zero and one and the sum of the α_i's is equal to one.

3 SPEAKER VERIFICATION SYSTEM

The speaker verification system used in this paper is known as the T-NETIX *SpeakEZ Voice Print*^{SM} system. This system is text dependent and utilizes sub-word NTN and GMM models, along with vocabulary-independent password selection and data fusion. The vocabulary-independent password selection is enabled through a technique known as blind segmentation [13]. The blind segmentation algorithm will automatically determine the number of segments and segment boundaries for a password without the use of transcription information. The NTN and GMM scores for each subword are accumulated to form the phrase-level score for each model type.

Additionally, a leave-one-out strategy is deployed to utilize the data redundancy in addition to facilitating threshold selection. Basically, for N enrollment repetitions of a password, there will be N separate models. Each model is trained with $N-1$ repetitions with a different repetition "left-out" for each model. The left-out repetition can then be applied to the model to yield an unbiased target speaker score that can be used in setting the threshold for speaker acceptance/rejection.

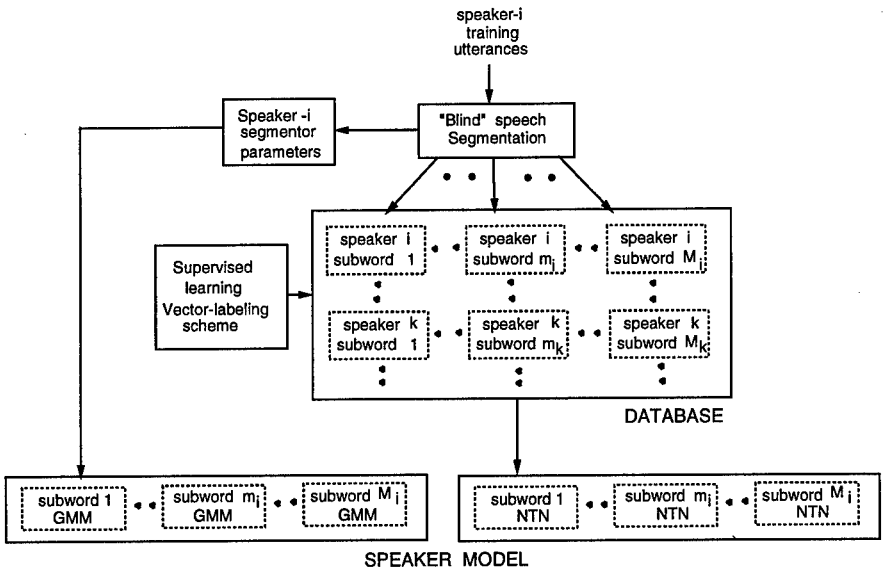


Figure 1: Training a speaker model

The procedure to train a model for a given speaker is illustrated in Figure 1. The multiple repetitions of the speaker's password are used by the segmentation module to estimate the number of subwords in the password along with the subword boundaries. The mean vector and diagonal covariance matrix of the subword segments are obtained as by-products of the segmentation module. These are used as the GMM component of the speaker model. For each subword segment of the password, a NTN model is also trained. The *closest* subword segments from other speakers who are already enrolled in the database are used as non-target data for training these subword NTN models.

The procedure to verify a claimed identity is illustrated in Figure 2. The given testing utterance is segmented to the optimal number of segments determined during training. The subword segment vectors are scored using the appropriate subword NTN and GMM models. The scores of these subword segments are averaged and a composite score for the entire phrase is obtained. The phrase-level NTN and GMM scores are then fused together using the linear opinion pool. We have performed experiments that did not show any advantages by combining this information at the subword level. If multiple models are obtained during training using the leave-one-out method, then all these models are scored in the above manner. These model scores are averaged to yield the final output score.

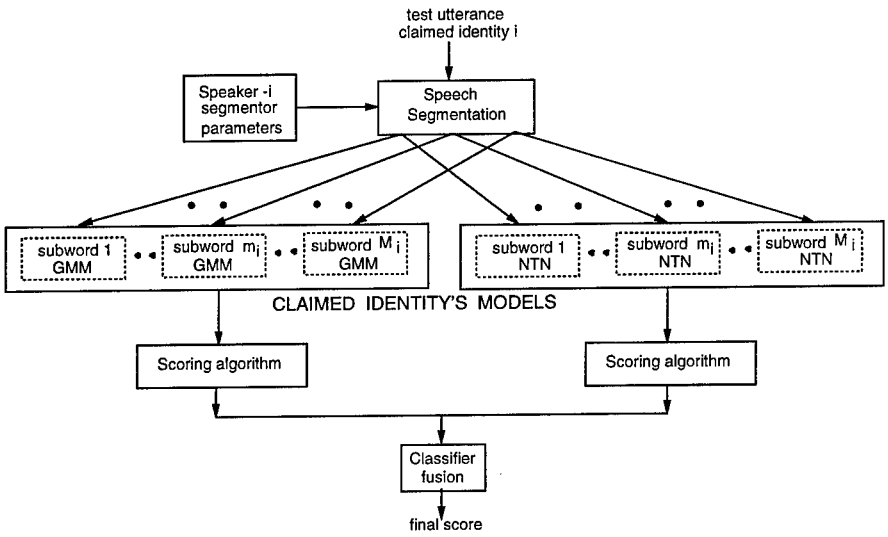


Figure 2: Testing a claimed identity

4 EXPERIMENTAL RESULTS

The T-NETIX *SpeakEZ Voice Print*^{SM} system is evaluated with three toll quality speech corpora that were collected by T-NETIX. The first database is known as the "names" database. The names database consists of 10 male target speakers, each with three enrollment utterances of their full name. The imposter attempts are comprised of the remaining nine speakers and all use the correct password. The second database is known as the "open sesame" database. This database consists of 56 enrolled speakers and 47 separate non-target speakers. Each speaker enrolled with the phrase "open sesame", hence, this scenario reflects a fixed-text situation. The third database is known as the "cellular" database. This database is also a fixed-text application that uses the password "Al Capone" for all speakers. This database was collected using cellular phones and consists of 26 evaluation speakers and and 15 non-target speakers. The aspects of each database are summarized in Table 1. The non-target speakers column in Table 1 refer to the *development* set that is used during training of a speaker model. To avoid bias in the results, the development speakers are not used as imposters during the actual testing. The *evaluation* speakers are used to measure the actual system performance.

The first experiment evaluates the system equal error rate as a function of the number of segments. Generally, the system computes the number of segments per password, but in this case, we have forced the number of segments to be constant for all speakers. The results of this experiment as performed on the names database are shown in Figure 3. It is clear from Figure 3 that the GMM requires several segments before the performance

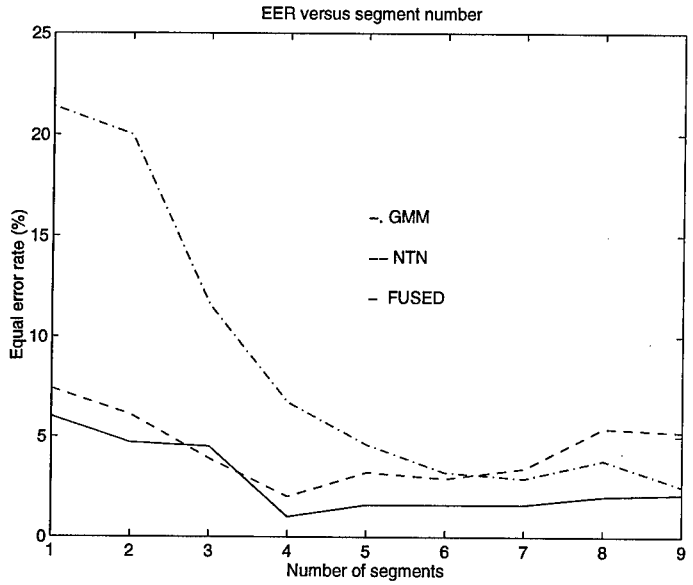


Figure 3: EER versus number of segments

starts to become competitive. The performance of the NTN, however, starts to degrade as the number of segments increases beyond four or five segments. This is due to the fact that the number of data samples per NTN decreases as the number of segments increases. Hence, for the NTN the lack of data starts to overcome the benefits of decomposing the acoustic space of the password.

The next experiment evaluates the equal error rate as a function of alpha for the linear opinion pool method of data fusion. The system uses a variable number of segments per speaker. The results of this experiment for the names database are shown in Figure 4. Here, it can be seen that the individual performance of the GMM and NTN is 3.2% and 3.4%, respectively. However, by combining the results of these methods, the EER can be reduced to 1.6%.

This experiment was also evaluated with the "Open Sesame" and "cellular" database and the results for these experiments are shown in Figures 5 and 6, respectively. The results for the "Open Sesame" database show the individual performance of the NTN and GMM to be 1.6% and 2.3%, respectively, whereas the performance of the fused output is 0.9%. For the cellular "Al Capone" database the individual performance of the NTN and GMM is 11.8% and 10.2%, respectively, while the performance of the fused output is 8.2%.

The experimental results for T-NETIX's *SpeakEZ Voice Print*^{SM} system are tabulated for the "names", "Open Sesame" and "cellular" databases in Table 1. The results in this table reflect the fusion results when $\alpha = 0.5$.

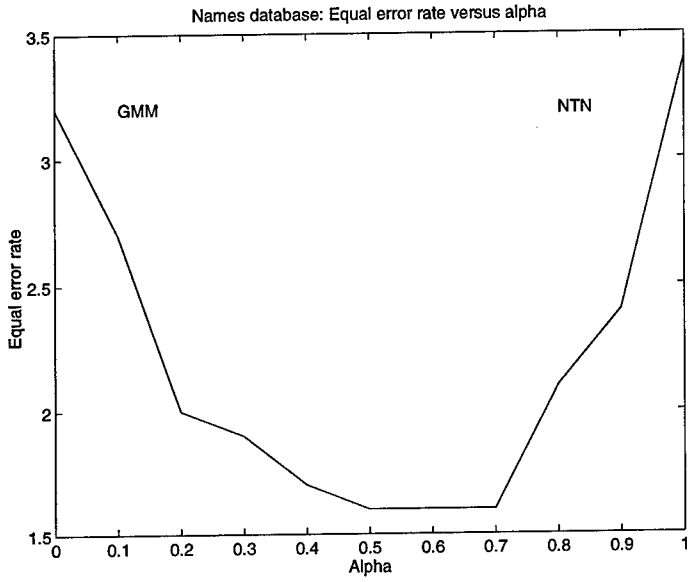


Figure 4: Linear opinion pool for names database

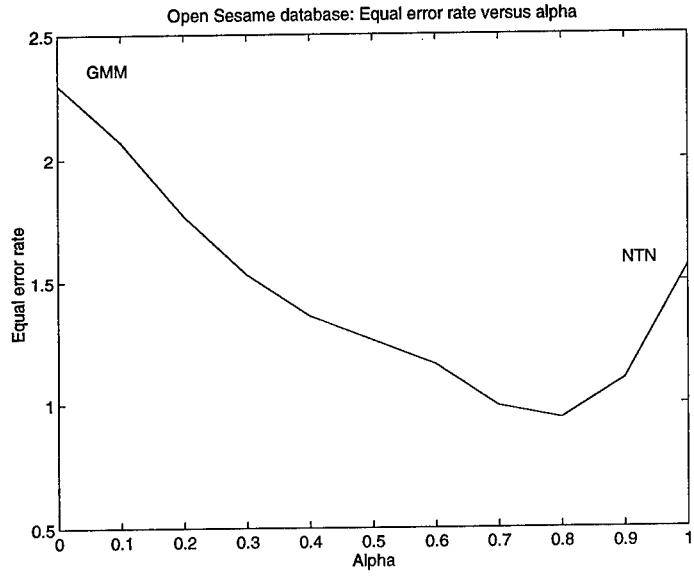


Figure 5: Linear opinion pool for "Open Sesame" database

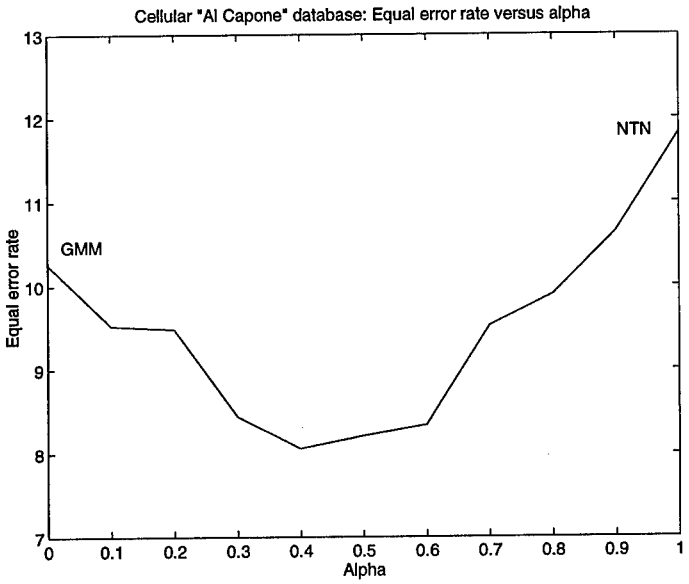


Figure 6: Linear opinion pool for "Cellular" database

5 CONCLUSION

The T-NETIX *SpeakEZ Voice Print*^{SM} system is evaluated for several text-dependent speaker verification tasks. These include applications in both cellular and landline environments. The T-NETIX *SpeakEZ Voice Print*^{SM} system does not have any constraints on the vocabulary from which the password is selected. This is accomplished through the use of sub-word neural tree networks and a blind segmentation algorithm that does not require phonetic label information. In addition, the system utilizes concepts within data fusion to capitalize upon different modeling approaches whose errors are uncorrelated. The data fusion techniques are found to reduce the error rate by a factor of two for the landline databases. The error rate for the cellular database is reduced by 20%. The error rates for the landline and cellular databases

Password text	# development/ evaluation speakers	# true/imposter trials	Performance (EER)
"Open Sesame"	47/56	195/11,229	1.3 %
Own full name	80/10 males	100/450	1.6 %
"Al Capone"	15/26	273/6825	8.2 %

Table 1: Performance for the *SpeakEZ Voice Print*^{SM} system

are roughly 1-2% and 8%, respectively. We find these results very encouraging given the constraints of limited training repetitions, short enrollment utterances, and unconstrained vocabulary for password selection.

References

[1] D. Naik. Pole-filtered cepstral mean subtraction. In *Proceedings ICASSP*, pages 157–160, 1995.

[2] A. Higgins, L. Bahler, and J. Porter. Speaker verification using randomized phrase prompting. *Digital Signal Processing*, 1:89–106, 1991.

[3] C.S. Liu, C.H. Lee, B.H. Juang, and A.E. Rosenberg. Speaker recognition based on minimum error discriminative training. In *Proceedings ICASSP*, pages 325–328, 1994.

[4] K.R. Farrell, R.J. Mammone, and K.T. Assaleh. Speaker recognition using neural networks and conventional classifiers. *IEEE Trans. Speech and Audio Processing*, 2(1), part 2, 1994.

[5] A. Sankar and R.J. Mammone. Growing and pruning neural tree networks. *IEEE Trans. on Computers*, C-42:221–229, March 1993.

[6] K.R. Farrell. Text-dependent speaker verification using data fusion. In *Proceedings ICASSP*, 1995.

[7] H. Liou and R.J. Mammone. Text-dependent speaker verification using sub-word neural tree networks. In *Proceedings ICASSP*, 1995.

[8] M. Sharma and R.J. Mammone. Subword-based text-dependent speaker verification system with user selectable passwords. In *Proceedings ICASSP*, 1996.

[9] K.R. Farrell. Discriminatory measures for speaker recognition. In *Proceedings of Neural Networks for Signal Processing*, 1995.

[10] D. Reynolds. Speaker identification and verification using Gaussian mixture models. *Speech Communications*, 17:91–108, August 1995.

[11] J.A. Benediktsson and P.H. Swain. Consensus theoretic classification methods. *IEEE Trans. on Systems, Man and Cybernetics*, 22(4):688–704, 1992.

[12] L. Xu, A. Krzyzak, and C.Y. Suen. Methods of combining multiple classifiers and their applications to hand-written character recognition. *IEEE Trans. on Systems, Man and Cybernetics*, 23(3):418–435, 1992.

[13] M. Sharma and R.J. Mammone. Blind speech segmentation: Automatic segmentation of speech without linguistic knowledge. In *Proceedings ICSLP*, 1996.

A CHAOTIC ANNEALING NEURAL NETWORK AND ITS APPLICATION TO DIRECTION ESTIMATION OF SPATIAL SIGNAL SOURCES

Ying Tan[1], Chao Deng[2], and Zhenya He[1]
[1]Department of Radio Engineering, Southeast University
Nanjing 210096 P.R. China
[2]Department of Computer Science, University of Science and Technology of China
Hefei 230026 P.R. China
E-mail: dengchao@cs.ustc.edu.cn

Abstract - A chaotic annealing neural network model based on transient chaos and dynamic gain is proposed for solving optimization problems with continuous-variables such as maximal likelihood estimation of spatial signal sources in this article. Compared to conventional neural networks only with point attractors, the proposed neural network has richer and more flexible dynamics which are expected to have higher ability of searching for globally optimal or near-optimal solutions. After going through an inverse-bifurcation process, the neural network gradually approaches to a conventional Hopfield neural network starting from a good initial state. Numerical simulations show both the effectiveness escaping from local minima and the ability solving for nonlinear maximal likelihood estimation of spatial sources of the proposed network.

I. INTRODUCTION

In many branches of science and technology one often encounters difficult optimization problems which have intractable computational complexity[1]. For these problems there is a large but finite set of possible solutions, among which we desire to find the one which globally minimizes the cost function involved. Typical examples are the knapsack problem and traveling salesman problem (TSP). Through the pioneer work in [2], the collective computational properties of the Hopfield neural network (HNN) for seeking a stable equilibrium can be utilized in solving many difficult optimization problems. The main difficulty in solving many actual optimization problems using HNN is that the network tends to become trapped in local minima due to its gradient descent dynamics. To avoid getting stuck in local minima[3], both stochastic simulated annealing (SSA)[4] approand deterministic simulated annealing (DSA) techniques have been proposed and are combined with

0-7803-4256-9/97/$10.00 ©1997 IEEE

neural networks. Typical examples of neural networks with SSA function are Boltzmann machines and Gaussian machines. On the other hand, various DSA approaches, such as hardware annealing (namely gain sharpening)[5] in the Hopfield networks and cellular neural networks and mean field approximate annealing[6], have been proposed and applied to neural networks.

Recently, a number of artificial neural networks with chaotic dynamics have been extensively investigated because of their more complex neurodynamics. Unlike the conventional networks only utilizing gradient descent dynamics, the neural networks with chaotic dynamics have richer and far-from equilibrium dynamics with various coexisting attractors, not only of fixed points and periodic points but also of strange attractors. This kind of complicated neurodynamics is a promising technique for information processing and optimization. In particular, an intriguing property of chaotic neural network to move chaotically over fractal structure in the phase space may be an efficient heuristic method searching for global optimal or near global optimal point, avoiding getting stuck at local minima. The maximum difficult happened for the use of the chaotic characteristics is to decide when to terminate the chaotic dynamics, or how to harness chaotic behavior for convergence to a stable equilibrium point corresponding to an acceptably near-optimal state.

In order to make full use of the advantages of both chaotic neurodynamics and conventional gradient descent (or convergent) neurodynamics, this article proposes a neural network model with transient chaos and dynamic gain for nonlinear optimization solving problem. This neural network is expected to have higher ability of searching for the global-optimal or near-optimal solution. The characteristics of the proposed network are analyzed and examined by numerical simulations in details.

II. A NEURAL NETWORK WITH TRANSIENT CHAOS AND TIME-VARIANT GAIN

It is well-known that Hopfield network with continuous-time or asynchronously discrete-time state transitions guarantee convergence to a stable equilibrium solution but suffer from local minimum problems. Since the chaotic neural network is of richer and more flexible neurodynamics whose running region is only a fractal structure in the phase space and may be used to efficiently escape from local minima problem in chaotically. Therefore, in order to obtain a global optimal or near-optimal convergent solution for nonlinear optimization, we ingeniously combine the chaotic dynamics with convergent dynamics and propose a new chaotic neural network model based on transient chaos and time-variant gain (NNTCTG), as defined below:

$$x_i(t) = \frac{1}{1+e^{-y_i(t)(1+\varepsilon_i(t))}} \tag{1}$$

$$y_i(t+1) = k y_i(t) + \alpha\left(\sum_{j=1, j\neq i}^{n} w_{ij} x_j(t) + I_i \right) - z_i(t)\left(x_i(t) - I_0\right) \quad (2)$$

$$z_i(t+1) = (1-\beta) z_i(t) \quad (3)$$

$$\varepsilon_i(t+1) = (1-\gamma)\varepsilon_i(t) \qquad (i = 1,2,\cdots,n) \quad (4)$$

where x_i , y_i and I_i are output, internal state and input bias of neuron i, w_{ij} connection weight from neuron j to neuron i, α = positive scaling parameter for inputs, k = damping factor of nerve membrane $(0 \le k \le 1)$, $z_i(t)$ = self-feedback connection weight $(z_i(t) \ge 0)$, β = damping factor of the time-dependent $z_i(t)$, $(0 \le \beta \le 1)$, $\varepsilon_i(t)$ = gain parameter of the output function $(\varepsilon_i(t) \ge 0)$, γ = damping factor of the time-dependent $\varepsilon_i(t)$, $(0 \le \gamma \le 1)$.

In (1)-(4), if $\varepsilon_i(t)$ equals to a big positive constant ε_i and $z_i(t)$ is a positive constant z_i and $I_0 = 0$, then the NNTCTG is reduced as the chaotic neural network (CNN) proposed in [7]. So the NNTCTG is regard as a generalized CNN and is expected to has some similar chaotic phenomenon of complicated bifurcation structures with CNN. By introducing the time-dependent variables $z_i(t)$ and $\varepsilon_i(t)$, the chaotic dynamics of NNTCTG can be reasonably harnessed and highly accurate steady-state solution for nonlinear optimization problem can be obtained as well as long as we choose appropriately the parameters in NNTCTG. This will be shown by numerically next section.

The term $z_i(t)\left(x_i(t) - I_0\right)$ in (2) is related to inhibitory self-feedback or refractoriness and is the main factor generating chaotic phenomenon. It can be shown that NNTCTG actually has transiently chaotic dynamics which eventually converges to a stable equilibrium point through successive bifurcations like a route of reversed period-doubling bifurcations, with the temporal evolution of $z_i(t)$ and $\varepsilon_i(t)$ by in (3). Variables $z_i(t)$ and $\varepsilon_i(t)$ corresponding to the temperature in usual stochastic simulated annealing process in exponential cooling schedule harness the chaotic behavior for convergence and the speed of reversed bifurcation. Actually, the damping of $z_i(t)$ and $\varepsilon_i(t)$ produces successive bifurcations so that the neurodynamics eventually converge from strange attractors to a stable equilibrium point.

Comparing with HNN, the NNTCTG has an additional nonlinear time-dependent damping term. As the self-feedback connection weights $z_i(t)$ tend toward zero with

time evolution in the form of $z_i(t) = z_i(0)e^{-\beta t}$, the NNTCTG defined in (1)-(4) eventually reduce to the continuous-time HNN which had been extensively used to solve many difficult optimization problem in many domains. On the other hand, with the temporal evolution of gain parameter $\varepsilon_i(t)$ simultaneously, the convergent NNTCTG after a reversed period-doubling bifurcation process gradually approaches to a HNN with desired gain value, which is very suitable for accurate nonlinear optimization solving of continuous-variables.

According to above discussions, it is clear that the procedure of NNTCTG in solving for nonlinear optimization problem is able to be divided into two phases: chaotic bifurcation phase and gradient convergent phase. In the first phase, a complicated and rich chaotic bifurcation process is created by big values of refractoriness and gain for the network to escape from local minima, whose mechanics can be regarded as a kind of DSA and called chaotic simulated annealing (CSA). After that, a good initial state at a neighborhood of globally optimal solution is provided for the gradient descent dynamics of the second phase of NNTCTG so that the network can easily reach the global optimal or near-optimal solution of the problem.

III. NEURODYNAMICS ANALYSIS OF NNTCTG

In order to analyze and examine the nonlinear dynamics of the proposed NNTCTG, we use it to solve a concrete optimization problem whose objective function is defined as

$$E(x_1,\ x_2) = (x_1 - 0.7)^2\left((x_2 + 0.6)^2 + 0.1\right) + (x_2 - 0.5)^2\left((x_1 + 0.4)^2 + 0.15\right). \quad (5)$$

Point (0.7, 0.5) is the global minimum while the points (0.6, 0.4), (0.6, 0.5) and (0.7, 0.4) are local minima in the landscape of energy function of (5).

Let us set the values of the network parameters in (1)-(4) as

$$k = 1.0;\quad \varepsilon(0) = [230,\ 230];\quad I_0 = 0.5;\quad z(0) = [0.082,\ 0.082]. \quad (6)$$

By Letting $\sum_{j=1}^{m} w_{ij}x_j + I_i = -\partial E/\partial x_i$, $(i = 1,2)$, the objective function of the concrete optimization problem can be transformed as the energy function of the corresponding NNTCTG. Through this transformation our proposed NNTCTG can be formed to solve this kind of optimization problems and can evolve from a given initial state.

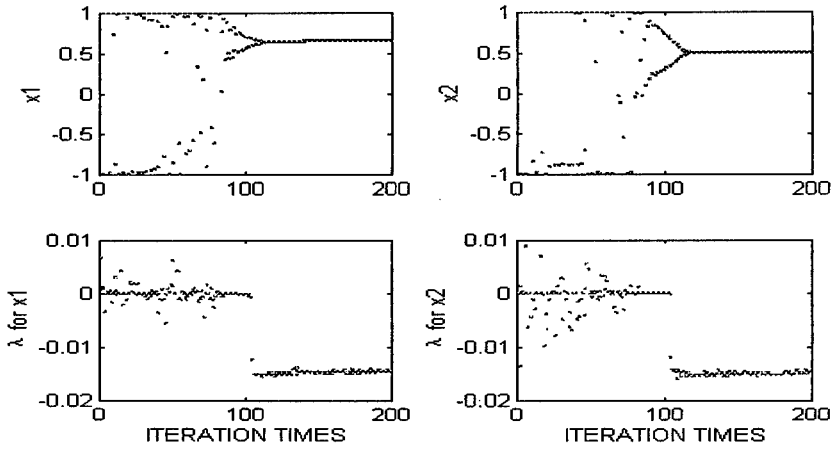


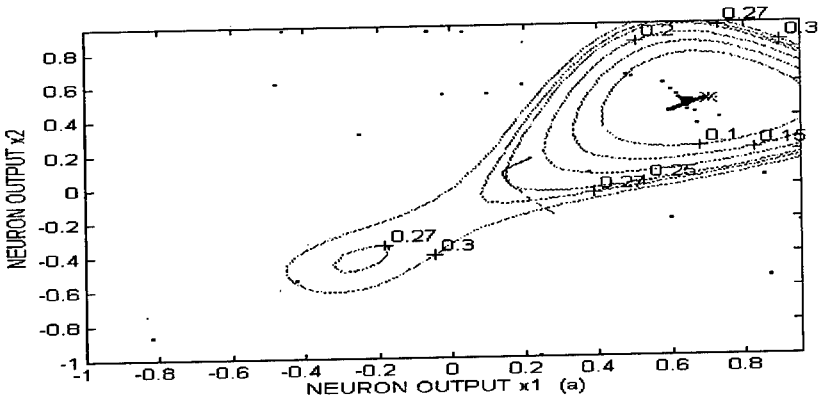
Fig.1 Time evolutions of X(t) and the Lyapunov exponent λ in the dynamics of the NNTCTG

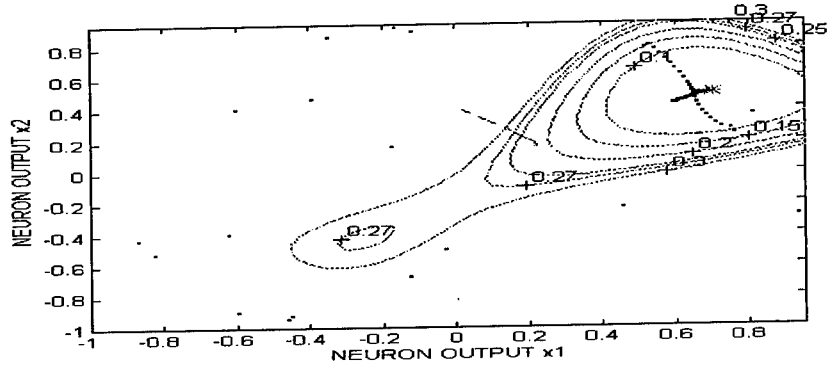
Fig.1 shows the time evolutions of $x_1(t), x_2(t)$ and their Lyapunov exponent λ, which are calculated by (1)-(4) with β =0.01, γ =0.01 and α =0.015 and given initial state y(0)=[-0.2828, -0.0461].

The Lyapunov exponent λ for $x_1(t)$ and $x_2(t)$ is calculated by

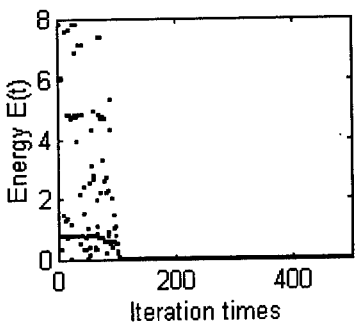
$$\lambda_i = \lim_{m\to\infty} \frac{1}{m} \sum_{k=0}^{m-1} ln \left| \frac{dx_i(k+1)}{dx_i(k)} \right|, \qquad i = 1, \quad 2 \tag{7}$$

The positive values of λ in Fig.1 indicate that NNTCTG actually have chaos during the first 110 iterations for $x_1(t)$ and $x_2(t)$ in this problem. After that, the network enters into its gradient convergent stage since the Lyapunov exponents are negative values. As $z_i(t)$ and $\varepsilon_i(t)$ are damped in exponential schedule simultaneously, Fig.1 shows clearly that the neuron outputs $x_1(t)$ and $x_2(t)$ gradually transit from chaotic behavior to fixed (or steady) values through a reversed period-doubling bifurcation process. In other words, the proposed NNTCTG has transiently chaotic dynamics and almost coincides with the HNN's dynamics when the values of time-dependent $z_i(t)$ and $\varepsilon_i(t)$ decreases enough. This supports our discussion last section. The fixed output of the network is just the global minimizer (0.70, 0.50) after 185 iterations.

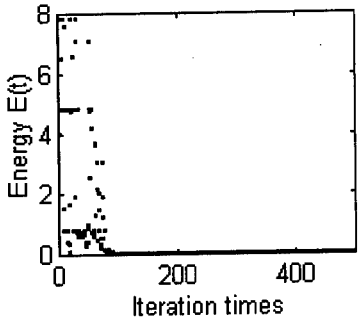




(b)



(c)



(d)

Fig.2 Multiple minima in energy function of (5) for two-neuron NNTCTG. (a)(b) Energy contour and trajectories of network outputs from initial states (0.5631,-0.3461) and (0.0337,0.8453), respectively. (c)(d) Corresponding energy functions E(t) during network evolutions for the two cases.

Fig.2 shows contour plots of the energy function and trajectories of output values of NNTCTG when network parameters are set as above except for the initial states (0.5631,-0.3461) and (0.0337,0.8453). The outputs initially located at a=(-1.0, -1.0) and b=(0.9992,1.0) follow the individual trajectories when the network is allowed to evolve. After a shortly bifurcation process, the steady-state outputs of the network for location a and location b are both (0.7, 0.5) which corresponds to the global minimum. It is also shown that the first stage of each trajectory is a temporal process of reversed period-doubling bifurcations whose searching regions in the state space shown by points in Fig.2(a) and (b), which can be at anywhere in whole state space in unpredictable or chaotic manner, is restricted to a small possibly fractal structure whose volume should be zero with respect to the Lebesgue measure of the whole state space. So CSA is much more efficient in computation than the famous SSA which needs to search the whole state space. Fig.2(c) and (d) also show that the energy functions corresponding to the two trajectories are also follow the same two phases, namely a fixed value (global minimum) is approached in gradient descent manner after a reversed bifurcations .

When we use the HNN to solve the problem with same conditions, the initial location a and b result in the local minimizer in the steady-state outputs whose trajectories are also shown in Fig.2 by dashed-line. From this attempt, it is clearly seen that the proposed NNTCTG with intrinsic CSA mechanics has much stronger ability searching for global optimal or near-optimal solution of nonlinear optimization than conventional HNN.

In the sequel we vary β and γ to investigate the dynamics of the NNTCTG while other parameters are fixed as in (6).

Fig.3 shows the time evolutions of neuron output $x_1(t)$ with different values of damping β and γ. As shown in Fig.3(a) and (c), with γ =0.001 or 0.005 and β =0.01, the parameter γ controls the bifurcation speed of transient chaos and the accuracy of the steady-state solutions. On the other hand, as was shown in Fig.3(b) and (d), when γ =0 and β =0.01 or 0.1 , after shortly bifurcation, the output of $x_1(t)$ becomes an oscillation and can not be stabilized at a fixed point at least at 2000 iterations done for the problem. In other words, the NNTCTG without time-variant gain can not be used to solve nonlinear optimization with continuous variables. Therefore, the damping factor γ of time-variant gain is a key parameter in NNTCTG which not only governs the bifurcation speed of the transient chaos and controls the solving accuracy of the network, but also trades off the gain requirements of both transiently chaos and steady-state HNN. In addition, the coefficient α is a balance parameter between the chaotic dynamics contributed by time-dependent term in (2) and convergent dynamics by the gradient term of energy function in (2), which should be chosen appropriately in practical applications.

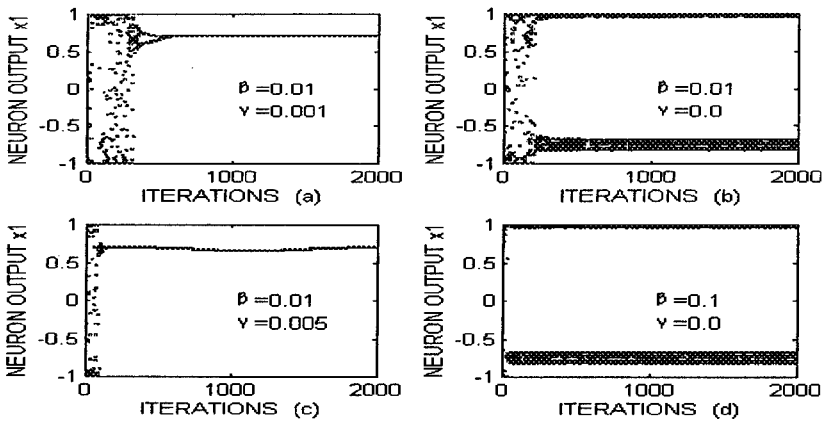


Fig.3 Trajectories of neuron x1 with different values of damping factors of β and γ

In order to further examine the ability of obtaining global and near-global solution for complex nonlinear optimization problem, we compute (5) with 10000 different initial states and list the results in Table 1. For convenient comparison, we also give the statistical results of HNN for the same conditions in Table 1. It can be seen that NNTCTG is always able to obtain the global optimal solution every time, but HNN can obtain the global optimal solutions for only 238 times because of its greedy gradient descent manner. Average iteration for convergence of NNTCTG is 195 iterations which is about one quarter time than that of HNN.

TABLE 1 RESULTS OF 10000 DIFFERENT INITIAL CONDITIONS ON OPTIMIZATION PROBLEM AS (6) BY NNTCTG AND HNN

Neural Network Model	NNTCTG	HNN
Rate of global minima (%)	10000(100%)	238(2.44%)
Rate of local minima (%)	0(0%)	9762(97.46%)
Average iterations for convergence	195	749

IV APPLICATION TO DOA ESTIMATION OF SPATIAL SOURCES

Recently highly accurate direction of arrival (DOA) estimations of spatial signal sources have been studied extensively and find applications in radar, communications, sonar, geophysical imaging and so on. Typical techniques are maximum likelihood (ML), MUSIC, minimum variance, propagating operator and ESPRIT, etc.. Although the ML method among them provides an optimum solution, it is not as prevailing as

so called suboptimal methods mainly because of its high computational complexity [8,9] for optimizing a nonlinear likelihood function expressed as

$$\min_{\theta_i,\varphi_i,S_i(l)} \sum_{l=1}^{M}\sum_{j=1}^{N}\left\{X_j(l)-\sum_{i=1}^{P}S_i(l)\cos\left[\frac{2\pi d}{\lambda}(j-1)\sin\theta_i+\varphi_i\right]\right\}^2 \tag{8}$$

where d is the interval between array units, λ is signal frequency, P denotes the number of signal sources, N is the number of array units, M is the number of flash shots. $X_j(l)$ represents the received data of jth unit in lth flash shot. Variable θ_i is the direction of the ith signal source, φ_i is the initial phase and $S_i(l)$ is the amplitude of signal sources.

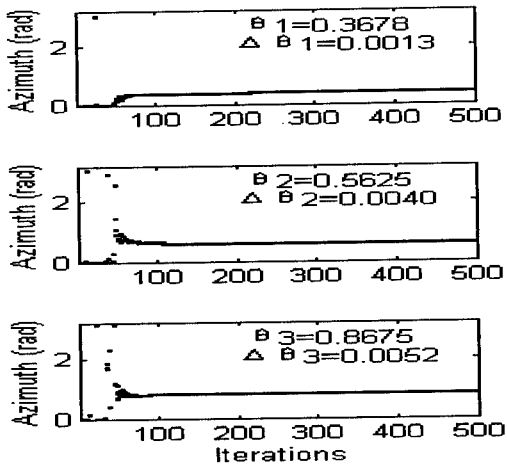


Fig.4 Trajectories of the network for estimating three signal sources

When to apply the parallel massive computational ability of Hopfield-like neural networks to this ML estimation in real time, it suffers from local minima problems. Here we use the proposed NNTCTG to solve this problem. Through appropriate transformation, equation (8) can be transformed as the energy function of NNTCTG which can be used to solve the ML DOA estimation in hand.

Example: DOA estimation for three narrow passband signal sources whose incoming angles are 21°, 32° and 50°, respectively. SNR is 20dB. N=5. The network parameters are chosen as $k=1.0$; $\varepsilon(0)=280$; $I_0=0.5$; $z(0)=0.082$. β =0.001, γ =0.04, α =0.009. Fig.4 shows the trajectories of the proposed network for the three spatial signal sources. It can be seen that the global optimal

solution, which is hard to be obtained by the conventional HNN-like methods, is easily reached by our neural networks.

IV. CONCLUSIONS

In this paper, we proposed a chaotic annealing neural network for accurately solving nonlinear optimization with continuous variables and applied it to ML estimation of spatial sources. It is an ingenious combination of chaotic dynamics and convergent dynamics whose intrinsic CSA mechanics has much higher ability to search for globally optimal or near-optimal solutions than conventional HNN and has higher computational efficiency than SSA. Numerical results have been given to examine and demonstrate the merit of the proposed neural networks.

Acknowledgments

This work was supported by the Climbing Programme - National Key Project for Foundamental Research in China, Grant NSC92097, and supported in part by Chinese Natural Science Fund.

References

[1]P.M. Pardalos and J.B. Rosen, **Constrained Global Optimization: Algorithms and Applications**, Berlin; Germany, Springer-Verlag, 1987.

[2]D.W. Tank, J.J. Hopfield, "Simple 'neural' optimization networks: an A/D converter, signal decision circuit, and a linear programming circuit," **IEEE Trans. Circuit Syst.**, vol.33, pp. 533-541, May 1986.

[3]Y. Tan, Z.Y. He, "Neural Network Approaches for the Extraction of the Eigenstructure," **Proceedings of the 1996 IEEE Workshop on Neural Networks for Signal Processing VI**, 1996, pp. 23-32.

[4]R.A. Rutenbar, "Simulated annealing algorithms: an overview," **IEEE Circuit Devices Mag.**, vol.5, no.1, pp. 19-26, Jan. 1989.

[5]S.H. Bang, B. J. Sheu, E.Y. Chou, "A hardware annealing method for optimal solutions on cellular neural networks," **IEEE Trans. Circuit and Syst.-II**, vol.43, no.6, pp. 409-421, Jan. 1996.

[6]C. Peterson, J.R. Anderson, "A mean field theory algorithm for neural networks," **Complex Systems**, 1, pp. 995-1019, Jan. 1987.

[7]K. Aihara, T. Takabe, M. Toyoda, "Chaotic neural networks," **Phys. Lett. A**, vol.144, no.6, pp. 333-340, 1990.

[8]J. Boheme, "Estimating the sources parameters by maximum likelihood and nonlinear regression," **Proc. of IEEE ICASSP'80**, pp. 307-310, 1980.

[9]R.O. Schmidt, "Multiple emitter location and signal parameter estimation," **IEEE Trans. on AP**, vol.34, no.3, pp.445-453,1986.

A NEURAL NETWORK EQUALIZER WITH THE FUZZY DECISION LEARNING RULE

Ki Yong Lee, *Sang-Yean Lee and **Stephen McLaughlin

Dept. of Electronics Engr., Changwon National Univ., Changwon, Kyungnam-Do, 641-773 KOREA E-mail: kylee@sarim.changwon.ac.kr
*** S.K. Telecom Ltd, Korea,**
**** Dept. of Electrical Engr., Edinburgh Univ., U.K.**

ABSTRACT

In this work, we propose an neural network equalizer with a fuzzy decision learning rule based on the generalized probabilistic descent algorithm with the minimum decision error formulation. Then, the neural network use the multi-layer perceptron. It is shown that the decision region overlapped by noise can be overcome by the use of a fuzzy decision learning rule based on the generalized probabilistic descent algorithm. We apply this algorithm to neural network equalizer with binary sequences in nonlinear distortion channel. Simulation results confirm that the fuzzy decision learning algorithm works more effectively than the hard decision learning algorithm when the learning patterns are not separable by high additive noise.

1. INTRODUCTION

In the digital communication equalization is an technique to reduce the influence of channels with nonlinear distortion such as amplitude and delay distortion. Application of neural-network techniques and fuzzy logic techniques to channel equalization leads to a better performance compared to the linear equalizer with inverse-filtering formulation [1-4]. The channel equalization based on neural network can be viewed as a classification problem in a geometric setting where an equalizer is constructed as a decision making device to reconstructed the transmitted symbol sequence.

The decision-based learning rule is effective for clearly separable decision boundary. When overlapping regions occurs due to the noise at the decision

This work was partly supported by Korea Research Foundation

0-7803-4256-9/97/$10.00 ©1997 IEEE

boundary, however, neural network equalizer suffer from poor discrimination. Such problem is well known in pattern and speech recognition [5,6]. To solve the problem, we need a technique based on a somewhat fuzzy decision appears to be more suitable.

In this work, we propose an neural network equalizer with a fuzzy decision learning rule based on the generalized probabilistic descent algorithm with the minimum decision error formulation [6]. This learning algorithm incorporates a penalty criterion into the conventional method with hard decision. Then, a penalty function treat the errors with equal penalty once the magnitude of errors exceeds certain threshold. The neural network for equalizer use the multi-layer perceptron (MLP). We apply this algorithm to neural network equalizer with binary sequences in nonlinear distortion channel.

2. NEURAL NETWORKS AS CHANNEL EQUALIZER

The transmitted data sequence $x(t)$ is assumed to be an independent sequence taking values from $\{-1, 1\}$ with an equal probability. The channel output $o(t)$ is corrupted by an additive noise. The task of the equalizer at sampling instant t is to produce an estimate of the input symbol $x(t-n)$ using the channel output vector $\boldsymbol{o}(t) = [o(t),\ldots,o(t-m+1)]$, where the integer m and n are known as the order and the delay of the equalizer, respectively. For describing a geometric formulation of the equalization problem, define

$$\begin{aligned} P_{m,n}(1) &= \left\{\hat{\boldsymbol{o}}(t) \in R^m \middle| x(t-n) = 1\right\} \\ P_{m,n}(-1) &= \left\{\hat{\boldsymbol{o}}(t) \in R^m \middle| x(t-n) = -1\right\} \end{aligned} \tag{1}$$

where R^m is the m-dimensional Euclidean space and, $P_{m,n}(1)$ and $P_{m,n}(-1)$ represent the two sets of possible channel noise-free output vectors $\hat{\boldsymbol{o}}(t)$ that can be produced from sequences of channel inputs containing $x(t-n) = 1$ and $x(t-n) = -1$, respectively. It is also clear that, if the states of $x(t)$, ..., $x(t-n)$ are finite, $\hat{\boldsymbol{o}}(t)$ can be only take finite values.

If the distribution of the noise is provided, the conditional density function of observing the channel output vector $\boldsymbol{o}(t)$ given $\hat{\boldsymbol{o}}(t) \in P_{m,n}(1)$and $\hat{\boldsymbol{o}}(t) \in P_{m,n}(-1)$, respectively, are completely specified. Denoting two conditional density functions as $b_1\left(\boldsymbol{o}(t), w^{-1}\right)$ and $b_{-1}\left(\boldsymbol{o}(t), w^{-1}\right)$, respectively, the equalizer

can be characterized by the function

$$\begin{aligned}\hat{x}(t-n) &= sgn\left(f_{de}(o(t))\right) \\ &= sgn\left(b_1\left(o(t), w^1\right) - b_{-1}\left(o(t), w^{-1}\right)\right)\end{aligned} \tag{2}$$

achieves the minimum bit-error-ratio (BER), where

$$sgn(y) = \begin{cases} 1, & y \geq 0 \\ -1, & y < 0 \end{cases}$$

represents a slicer.

The set

$$\left\{o(t) \in R^m \middle| f_{de}(o(t)) \geq 0\right\}$$

is known as the decision region of the optimal BER equalizer and the decision boundary of this equalizer consists of the set of points

$$\left\{o(t) \in R^m \middle| f_{de}(o(t)) = 0\right\}$$

When the decision boundary is clearly separable, the decision-based learning algorithm is effective to solve this equalization problem [1]. However, due to the noise the decision boundary often are not separable.

In decision boundary, the influence of noise is now investigated. The optimal boundary of NN equalizer with m=2 and n=0 for Gaussian white noise with SNR 15 dB and 5 dB are illustrated in Fig.1, where the region on the left half plane of the boundary is the optimal decision region. It is seen that noise clearly affects the decision region.

3. LEARNING RULE BASED ON THE FUZZY DECISION

One way of providing tolerance, for the non separable case, can be derived based on a somewhat fuzzy decision [7]. The fuzzy decision has different degrees of error associated with each decision, for example, marginally erroneous, erroneous, and extremely erroneous. The technique imposes a proper penalty function on all the 'bad' decisions as well as the 'marginally correct' ones. The final solution represents the best compromise in terms of the total

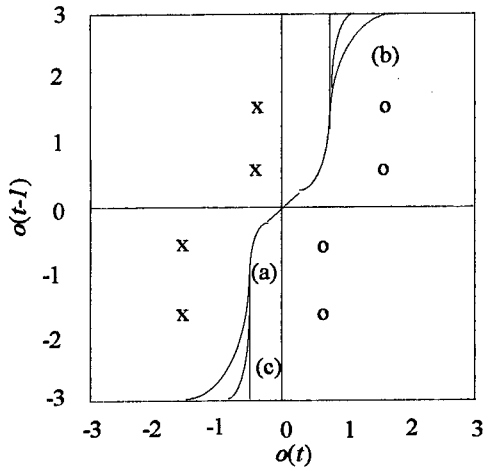


Fig. 1. Channel output points and optimal decision boundary. Channel $\hat{o}(t) = x(t) + 0.5x(t-1)$, equalizer order m=2 and lag n=0;

(a) optimal decision boundary,

(b) decision boundary from hard decision under 5 dB,

(c) decision boundary from hard decision under 15 dB.

penalty. In short, this allows 'soft' or 'fuzzy' decision, as opposed to the hard decision. To cope with 'marginal' training sequence, and to provide a smooth 'gradient' for training, the penalty must be a function of the degree of error. To derive the fuzzy decision training, firstly we can choose an appropriate discriminate function $g_l(o(t); w^l)$ as

$$g_l(o(t); w^l) = \left(d(t) - b_l(o(t), w^l)\right)^2, \; l = -1, 1 \qquad (3)$$

where in the training mode $d(t)=x(t-n)$ and during the data transmission $d(t) = \hat{x}(t-n)$.

The misclassification measure Q for a desired sequence belonging to the 1-decision region is defined as

$$Q(t) = -g_1(o(t); w^1) + g_{-1}(o(t); w^{-1}) \qquad (4)$$

The expected error as an objective criterion for weights w of MLP is defined as follows:

$$L(w) = E\left[J(Q(t))\right] \tag{5}$$

where a loss function $J(Q(t)) = \frac{1}{1+e^{-\alpha Q(t)}}$ is defined to evaluate the cost of the current decision and α is a positive constant for scaling.

The loss function provides a means to minimize the number of decision errors and will induce the learning rule to emphasize state located at a short distance from the decision boundary. Then, the loss function has four regions for fuzzy decision: (1) correct with satisfactory vigilance, (2) correct with vigilance to be improved, (3) error on which correction will be attempted, (4) error to remain uncorrected, as Fig. 2.

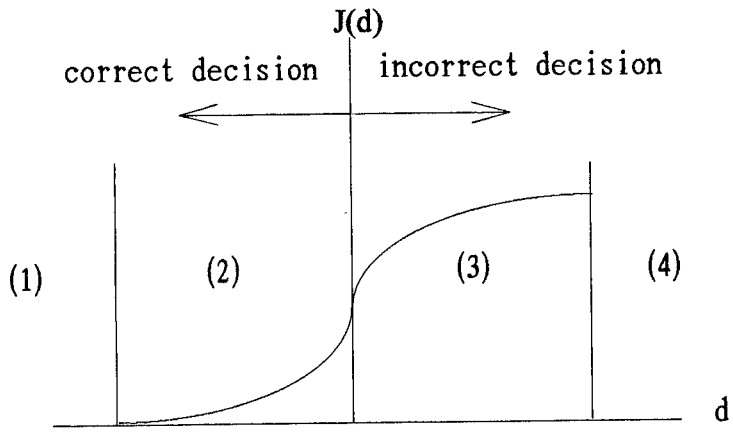


Fig. 2. Four regions for fuzzy decision.

The gradient descent search method can be applied to minimize the expected error of (4). Suppose that a current desired sequence is known to belong to decision region of $1; J(Q(t)) \in C^1$. Finally, we can obtain the fuzzy decision training rule as

Reinforced training:

$$w^1(t+1) = w^1(t) + \eta J'(Q(t)) \frac{\partial Q(t)}{\partial w^1(t)} \tag{6}$$

Antireinforced training:

$$w^{-1}(t+1) = w^{-1}(t) - \eta J'(Q(t)) \frac{\partial Q(t)}{\partial w^{-1}(t)} \tag{7}$$

where $J'(Q(t))$ is the derivative of the loss function evaluated at $Q(t)$.

In (6) and (7), weights are adjusted in proportion to the value of $J'(Q(t))$ and the maximum changes in the weights are happened when $Q(t)=0$, which means that the decision criterion is exactly in the boundary of region 1 and -1. So the more confusable those two models are, the reinforced training is carried out on the parameters of the correct model, while the more antireinforced training is carried out on the parameters of the incorrect model. Consequently, the discrimination between the correct model and the incorrect model will be increased.

3. EXPERIMENTAL RESULTS

In order to evaluate the performance of the proposed method, the equalization for nonlinear channel with noise is studied. The channel model is

$$o(t) = \hat{o}(t) + 0.2\hat{o}^2(t) + v(t), \quad v(t) \approx N(0, \sigma^2)$$

$$\hat{o}(t) = 0.3482x(t) + 0.8704x(t-1) + 0.3482x(t-2)$$

For $m=3$ and $n=0$, the 3-layer neural network (3-9-5-1) with 3-input and 1-output is used as equalizer. Under SNR 5 dB and 15 dB, Fig. 3 show the decision region formed by a NN equalizer with fuzzy decision learning algorithm and conventional decision learning algorithm, and optimal decision region. It can be seen that the decision region formed by the fuzzy decision is near that optimal decision region rather conventional decision.

We compare the bit error rates (BER) achieved by NN equalizer and NN equalizer with decision feedback (DF) [8] using the proposed method and conventional method for different SNRs and scale α of loss function. The equalizers are trained for the first 1000 points. Fig. 4 illustrates the BER performance averaged over 20 runs started from different random initial weights. It may be observed from Fig. 4 that the equalizer using the proposed method attains about 0.5-1.0dB improvement relative to the equalizer using the hard decision learning rule, although the correspondence between the curves is closet in the low noise situations.

Fig. 5 illustrate BER performance of NN equalizer with decision feedback using the fuzzy decision and hard decision. In NN equalizer with DF also, the proposed method performs the well performance in comparison with the hard decision, when the level of additive noise is high.

4. CONCLUSIONS

In this work, we propose an neural network equalizer with a fuzzy decision learning rule based on the generalized probabilistic descent algorithm with the minimum decision error formulation. It is shown that the decision region overlapped by noise can be overcome by the use of a fuzzy decision learning rule based on the generalized probabilistic descent algorithm. We applied the proposed method to equalize the nonlinear communication channel with noise. Simulation results confirm that the fuzzy decision learning algorithm works more effectively than the hard decision learning algorithm when the learning patterns are not separable by high additive noise situations.

REFERENCES

[1] Chen, S., Gibson, G.J., Cowan, C.F.N., and Grant, P.M. "Adaptive equalization of finite non-linear channels using multilayer perceptrons," *Signal Proc.*, 20, pp. 107-119, 1990

[2] Gibson, G.J., Siu, S., and Cowan, C.F.N., "The application of nonlinear structures to the reconstruction of binary signals," *IEEE Trans., Signal Proc.*, pp. 1877-1884, 1991

[3] L.X. Wang and J. Mendal, ""Fuzzy adaptive filters with application to nonlinear channel equalization," *IEEE Trans. Fuzzy Syst.*, vol.1, no.3, pp. 161-170, 1993.

[4] K.Y. Lee, "Complex fuzzy adaptive filter with LMS algorithm," *IEEE Trans. Signal Proc.*, vol.44, no.2, pp. 424-427, 1996.

[5] Duda, R.O. and Hart, P.E., *Pattern Classification and Scene Analysis.* New York: Wiely, 1973

[6] Juang, B.H. and Katagiri, S., "Discriminative learning for minimum classification," *IEEE Trans., Signal Proc.*, pp. 3043-3054, 1992

[7] J.S. Taur and S.Y. Kung, "Fuzzy-decision neural networks," In Proc. *IEEE ICASSP*, pp. I-577-I.580, 1993.

[8] S. Siu, G.J. Gibson, and C.F.N. Cowan, "Decision feedback equalisation using neural network structures and performance comparison with standard architecture," *IEE Proc.*, vol. 137, Pt.1, 4, pp.221-225, Aug. 1990.

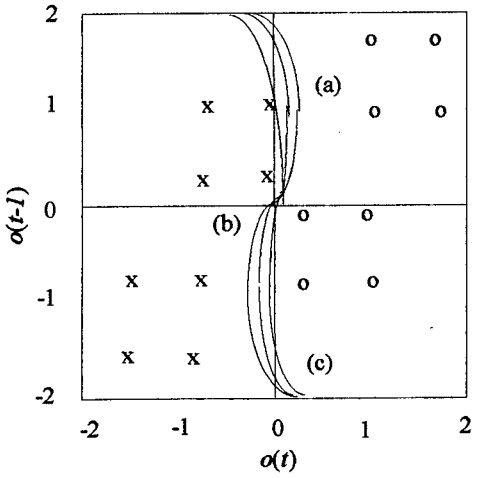


Fig. 3. Decision region under 10 dB; (a) optimal decision boundary, (b) boundary with hard decision , (c) boundary with fuzzy decision

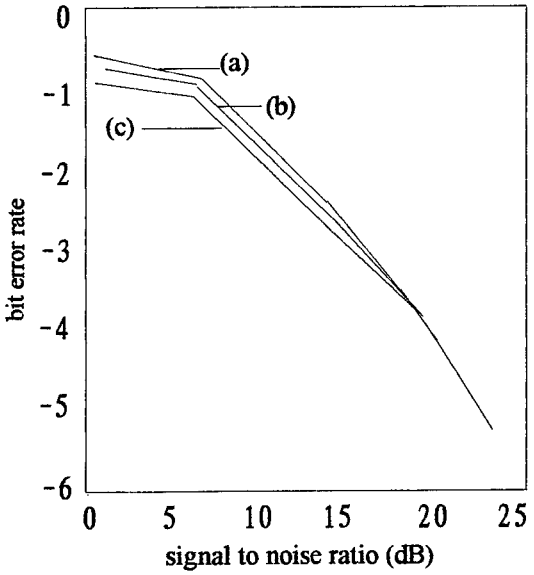


Fig. 4. Comparison of BER achieved by the MLP equalizer. (a) hard decision learning rule, (b) fuzzy decision learning ($\alpha = 10$), (c) fuzzy decision learning ($\alpha = 100$).

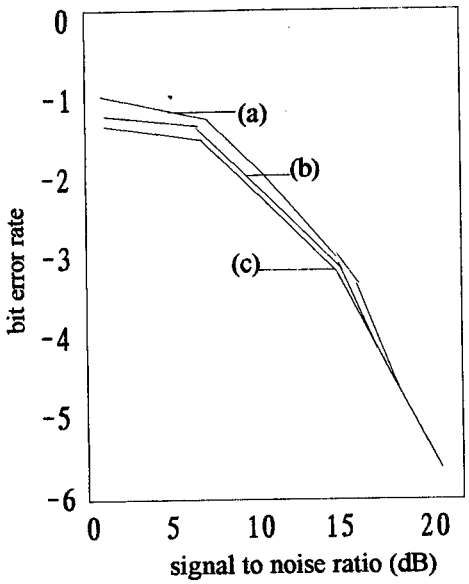


Fig. 5. Comparison of BER achieved by the MLP equalizer wirh decision feedback. (a) hard decision learning rule, (b) fuzzy decision learning ($\alpha = 10$), (c) fuzzy decision learning ($\alpha = 100$).

APPLICATION OF THE BLOCK RECURSIVE LEAST SQUARES ALGORITHM TO ADAPTIVE NEURAL BEAMFORMING

E.D.Di Claudio, R.Parisi, and G.Orlandi
INFOCOM Department
University of Rome " La Sapienza" Via Eudossiana 18 - I-00184 Rome, Italy.
email: parisi@infocom.ing.uniroma1.it, dic@infocom.ing.uniroma1.it
Ph.: +39-6-44585837, Fax: +39-6-4873300

ABSTRACT

Spatial beamforming using a known training sequence is a well-understood technique for canceling uncorrelated interferences from telecommunication signals [1]. Most of on-line adaptive beamforming algorithms are based on linear algebra and linear signal models. Anyway both in the transmitter amplifier and in the array receiver nonlinearities may arise, producing distorted waveforms and reducing the performance of the demodulation process. A nonlinear spatial beamformer with sensor arrays may use a neural network to cope with communication system nonlinearities.
In this work we show that a feedforward neural network trained with a LS-based algorithm may get the convergence in a time suitable to most applications.

1. INTRODUCTION

Signal processing by sensor arrays is sought as a technique for improving location parameter estimation (high resolution algorithms) and increasing the capacity of telecommunication links. The key feature of sensor arrays is that the number of processed sources and the Signal-to-Noise-Ratio (SNR) are limited in principle only by the system size [2].
In particular an array is able to create a spatial gain pattern with somewhat arbitrary shape by properly combining the outputs, according to a specified optimization criterion. Different patterns can be formed in parallel from the same received signals, enabling simultaneous demodulation and interference suppression in a multiple source environment.
If the beam computation is realizable by an on-line approach, the capacity of the communication system can be substantially improved. This requires adaptive algorithms able to converge to the steady-state solution before link parameters change. This is a serious problem in mobile communication systems, that may suffer also from the presence of nonlinearities along the signal path [3] and impulsive interferences, thus requiring a proper nonlinear treatment [4].
The signal model at the array output is represented by the classical linear equation [2]:

0-7803-4256-9/97/$10.00

$$\mathbf{x}(t) = \mathbf{A}\begin{bmatrix}\mathbf{s}(t)\\ \mathbf{i}(t)\end{bmatrix} + \mathbf{n}(t) \qquad (1)$$

where **x**(t) is the M-by-1 sensor snapshot vector at time t, **s**(t) is the ideal K-by-1 (K<M) signal vector, **A** represents the (unknown) M-by-Q array steering matrix [2], **i**(t) is the (Q-K)-by-1 interference (K≤Q<M) and **n**(t) is the additive background noise, uncorrelated with source and interference. All involved signals are assumed to be complex-valued (analytic signals) [1],[2].
Adaptive spatial beamforming often uses a preamble training sequence **y**(t) to recognize useful signals and cancel interferences by steering array nulls toward disturbances [1].
y(t) may be an analog local replica of the useful signal(s) **s**(t), or the sequence of modulated symbol vectors $\{\mathbf{a}_i, i=1,2,..,N\}$ which forms **s**(t) [3]:

$$\mathbf{s}(t) = \sum_{i=1}^{N} \mathbf{a}_i \mathbf{u}(t - iT) \qquad (2)$$

The goal is to choose the proper parameters of a function $F(\mathbf{x}(t))$ of the array output which best approximates **y**(t). In classical beamforming the function is linear and is expressed as the Hermitian product with a weight vector **w**:

$$\mathbf{y}(t) = \mathbf{w}^H \mathbf{x}(t) + \mathbf{e}(t), \qquad (3)$$

where **e**(t) is the approximation error.
Due to channel non-stationarity and communication efficiency requirements the adaptation process of **w** should be fast. The most widely known algorithms for on-line adaptation are based on the stochastic gradient descent, or Least Mean Squares (LMS) [5]. Anyway, the rate of convergence of LMS is linear [6] and is bounded by $(\rho-1)^2/(\rho+1)^2$, where ρ is the condition number of the Hessian matrix [6].
As a matter of fact, the Hessian matrix in narrowband adaptive beamforming coincides with the (scaled) spatial cross sensor correlation matrix (CSCM) of sensor outputs. Its condition number is of the same order of the array SNR and can be very high ($10^2 \div 10^9$) in telecommunication applications. For this reason linear beamformers frequently use methods based on Recursive Least Squares (RLS) to get higher rates of convergence with respect to classical gradient-based approaches [5].

2. NEURAL BEAMFORMING

In communication systems the adaptive array is part of a chain of blocks; most of them are intrinsically nonlinear or may exhibit undesired nonlinearities (amplifiers, mixers, clampers, ...). In order to cope with these nonlinearities, a

nonlinear beamformer should be employed. Feedforward multilayer neural networks may provide a solution to this problem [7] in a straightforward approach.
The input-output relationship of the proposed neural network beamformer with M sensors is:

$$\mathbf{y}(t) = F\{\mathbf{x}(t)\} + \mathbf{e}(t) \tag{4}$$

In this case the correct model for $\mathbf{x}(t)$ is related to $\mathbf{s}(t)$ by the equation:

$$\mathbf{x}(t) = \mathbf{A}G\left\{\begin{bmatrix}\mathbf{s}(t)\\ \mathbf{i}(t)\end{bmatrix}\right\} + \mathbf{n}(t) \tag{5}$$

In this formula G{.} is an unknown nonlinear M-by-Q matrix transfer function, that should be inverted by the neural beamformer.
Using the L_2 norm, the error functional J to be minimized is:

$$J = E\{trace[\mathbf{e}(t)\mathbf{e}^{H}(t)]\} \tag{6}$$

where $E\{.\}$ denotes the expectation operator over time, *trace*[.] is the matrix trace operator and $(.)^H$ indicates Hermitian transposition.
The neural beamformer realizes a nonlinear memoryless functional of $\mathbf{x}(t)$. A standard multilayer perceptron (MLP) can approximate arbitrary input-output relationships of this kind [7]. The input of the MLP are the sensor outputs, while $\mathbf{y}(t)$ contains the target signals.
The minimization of J may be accomplished by separating the real and the imaginary parts of all signals and using a standard backpropagation (BP) approach [7]. As an alternative, the complex neuron model can be used, which gives some benefits for the reduced number of free weights [5],[7].
Backpropagation is a stochastic gradient descent method and is characterized by the same limitations of LMS [7]. Faster second-order [6] convergence can be obtained with the Block Recursive Least Squares (BRLS) approach described in [8],[9]. In [9] BRLS is shown to be a Newton-type algorithm able to reach convergence with a very favorable numerical conditioning.
In this work we show by numerical simulations how a neural beamformer trained with the BRLS technique can be very effective in the presence of high levels of noise and interference, while detecting and recovering multiple signals of interest.

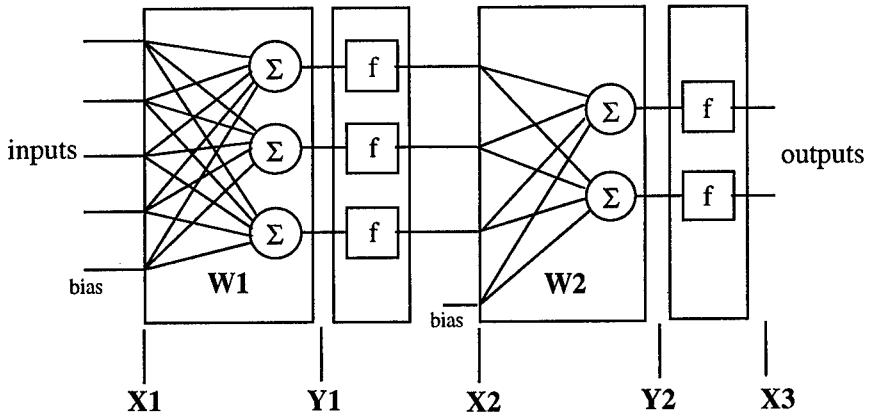


Figure 1: two-layer MLP.

3. SUMMARY OF THE BRLS ALGORITHM

The BRLS algorithm is an iterative learning procedure which solves an overdetermined linear system of equations for each layer of the network. With respect to similar algorithms, it minimizes the error functional at the linear summation nodes of the neurons (*descent in the neuron space* [9]), in contrast to the traditional optimization in the *weight space*.
At the n-th iteration and for the k-th layer we introduce the following matrices:

$$\mathbf{X}_k(n) = \begin{pmatrix} \mathbf{x}^T_{k,1}(n) \\ \dots \\ \mathbf{x}^T_{k,P}(n) \end{pmatrix} \mathbf{Y}_k(n) = \begin{pmatrix} \mathbf{y}^T_{k,1}(n) \\ \dots \\ \mathbf{y}^T_{k,P}(n) \end{pmatrix} \qquad (7)$$

where $\mathbf{x}^T_{k,p}(n)$ and $\mathbf{y}^T_{k,p}(n)$ are the input and output row vectors of the linear section of the MLP layer, in the presence of the p-th learning pattern (P is the length of the whole batch). The lengths of $\mathbf{x}^T_{k,p}(n)$ and $\mathbf{y}^T_{k,p}(n)$ are respectively (N_k+1) and N_{k+1}, being N_k the number of input units to the layer; $\mathbf{X}_1(n)$ contains the external inputs, while $\mathbf{X}_{L+1}(n)$ contains the global outputs of the net (see fig. 1 for the case L=2).
Matrices $\mathbf{Y}_k$ are computed in a forward-propagation step through the k-th layer, similar to that of standard BP [7]:

$$\mathbf{X}_k(n)\mathbf{W}_k(n) = \mathbf{Y}_k(n) \qquad (8)$$

while the passage through the nonlinearities is represented by the following:

$$\mathbf{X}_{k+1}(n) = [f(\mathbf{Y}_k(n)) \quad \mathbf{1}] \tag{9}$$

f(.) represents the activation function and **1** is the column of the bias inputs. At the first iteration weight matrices $\mathbf{W}_k$ are initialized at random.
The optimization in the neuron space is performed separately for each layer following the formula:

$$\hat{\mathbf{Y}}_k(n) = \mathbf{Y}_k(n) + \mathbf{D}_k(n) \tag{10}$$

where $\hat{\mathbf{Y}}_k$ is an estimation of the desired $\mathbf{Y}_k$ based on the *direction matrix* $\mathbf{D}_k$.
$\mathbf{D}_k$ may be chosen in several ways, depending on the method being adopted. The simplest choice is the negative of the *gradient matrix*:

$$\hat{\mathbf{Y}}_k(n) = \mathbf{Y}_k(n) - \eta \nabla_{\mathbf{Y}_k} E \tag{11}$$

where η is a proper *correction factor*. The derivatives of E w.r.t. the **y**'s are computed using formulas similar to those of BP [7].
Given the estimate $\hat{\mathbf{Y}}_k$, the new weight matrix for each layer is computed from the Least Squares (LS) solution of the following system:

$$\mathbf{X}_k(n)\mathbf{W}_k(n+1) = \hat{\mathbf{Y}}_k(n) \tag{12}$$

where in particular QR or SVD based algorithms can be used [5]. Formula (12) represents the general formulation of the class of LS-based learning algorithms; it consists in perturbing the matrix $\mathbf{Y}_k$ in order to recover the consistency of the system in the LS sense. This gives the new weight matrix $\mathbf{W}_k(n+1)$, to be used in the next forward-propagation step.
In order to stabilize learning in earlier steps, when weight are far from optimal values, a recursive implementation can be adopted, updating the solution by the classical on-line RLS-QR algorithm [5],[9]. In this case the forward and backpropagation phases act on the last block of snapshots [2]. A proper exponential forgetting factor λ can be used to discard the influence of older samples [5]. More details about the BRLS algorithm can be found in [9]. Here we point out that the numerical robustness of the algorithm is threatened by the severe ill-conditioning of matrix $\mathbf{X}_1$ which is just the square root of the array CSCM. However the use of a square root formulation keeps the condition number acceptably low ($10^4 \div 10^5$), allowing the use of the limited precision floating-point arithmetic offered by commercial DSP microprocessors.

4. EXPERIMENTAL TRIALS

In the proposed computer experiment a linear equispaced array of ten sensors, with an intersensor spacing of half wavelength, is used as receiver. The useful signals are two independent 8-QAM waveforms with a SNR of 10 dB w.r.t. each sensor, impinging from 8 and 25 degrees of azimuth, referred to the array broadside; the elevation is zero degrees for both sources. The signals of interest are distorted by a memoryless arctangent nonlinearity, which models the amplifier static transfer function. The interference is represented by a white Gaussian noise source, coming from a direction of 15 degrees, with a SNR of 20 dB. The array receiver gains fluctuate with a standard deviation of 1% of their nominal values during the experiment. The background noise is supposed to be Gaussian, white and isotropic [2]. The conditions of the experiment are recognized to be rather unfavorable since all coherent sources are within one array beamwidth [2], [3].
The neural network used in the experiment is a multilayer perceptron with 20 inputs, 8 hidden neurons and four outputs, with sigmoidal-type nonlinearities. Learning was performed with the BRLS algorithm on 100 epochs of 30 snapshots each. Several values for the forgetting factor λ were tried; in the described experiment λ=0.99 was used. The following figure shows the curve of the ratio of the target signal power (Mean Squared Signal, MSS) to the error power (Mean Squared Error, MSE) for each source of interest during the learning. The steady state solution is reached after about 35 epochs.
The almost monotonic shape of the learning curves demonstrates the ability of the BRLS algorithm to deal with ill-conditioned problems and to track the short time channel fluctuations that can be expected in real systems. Also remarkable is the insensitivity of the BRLS method to the starting guess of the network weights [9], which is essential for successful signal processing applications.

CONCLUSION

The recently introduced BRLS algorithm for fast training MLP networks allows the use of neural architectures in challenging multichannel DSP problems, characterized by severe ill-conditioning of the data matrix coupled with stringent requirements on convergence rate. The general approach described in [9] has a great flexibility in changes of the error functional and learning parameters, and may introduce several forms of weight regularization through system (12) [5]. We plan to apply the BRLS neural approach to combine space-time equalization of communication channels and nonlinear distortion correction.

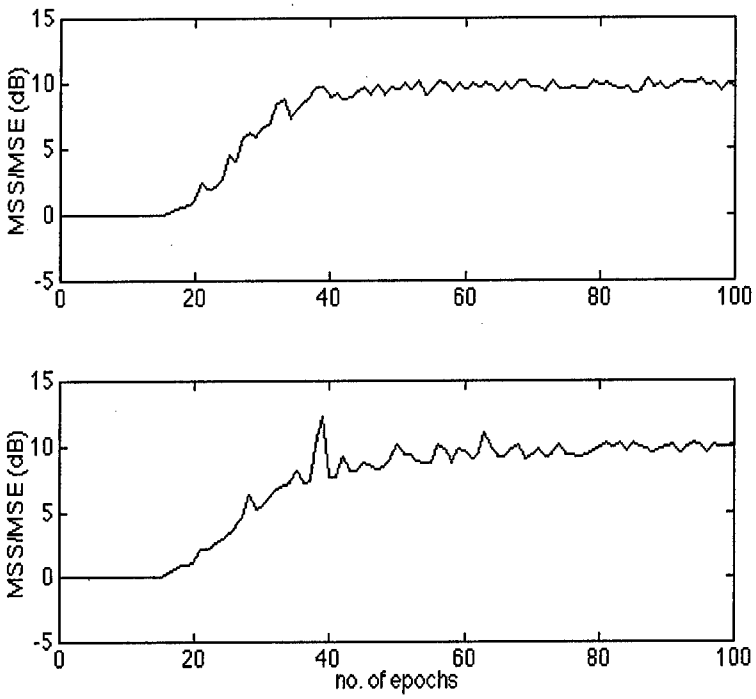


Figure 2: Learning curves for the two complex outputs.

REFERENCES

[1] D.H.Johnson and D.E. Dudgeon, *Array Signal Processing: Concepts and Techniques*, Prentice-Hall, 1993.

[2] R.O.Schmidt, "Multiple Emitter Location and Signal Parameter Estimation," *IEEE Trans. on Antennas and Propagation*, Vol. 34, No. 3, pp.276-280, March 1986.

[3] E.A.Lee, D.G.Messerchmitt, *Digital Communication*, Kluwer Academic Publishers, Boston, USA, 1988.

[4] L.P.Ammann, ``Statistically robust signal subspace identification," *Proc. of the Int. Conf. on ASSP*, ICASSP'90, pp 2711-2714, 1991.

[5] S.Haykin, *Adaptive Filter Theory,* Prentice Hall, 2nd ed. 1991.

[6] D.G.Luenberger, *Linear and Nonlinear Programming*, Addison Wesley,2nd ed. 1989.

[7] S.Haykin, *Neural Networks-A Comprehensive Foundation*, IEEE Press, 1994.

[8] E.D.Di Claudio, R. Parisi, G. Orlandi, "Block-recursive least squares technique for training multilayer perceptrons," *1994 World Conference on Neural Networks (WCNN'94)*, San Diego, May 26-29, 1994.

[9] R.Parisi, E.D. Di Claudio, G. Orlandi and B.D. Rao, "A generalized learning paradigm exploiting the structure of feedforward neural networks," *IEEE Trans. on Neural Networks*, vol.7, no.6, November 1996.

NONLINEAR COMMITTEE PATTERN CLASSIFICATION

Yu Hen Hu, Thomas Knoblock, and Jong-Ming Park
Department of Electrical and Computer Engineering
University of Wisconsin
Madison, WI 53706
Email: hu@engr.wisc.edu

ABSTRACT

Methods which combines outputs of multiple pattern classifiers to enhance the overall classification rate are studied. Specific attention is given to combination rules which are independent of the input feature vectors. Potential performance enhancement and limits of this so called stack generalization method are discussed. In particular, a phenomenum called "alias" is introced which gives an upper bound of the performance which can be achieved uisng stack generation for a given set of member classifiers. Experimentation using several machine learning data bases are reported.

I. INTRODUCTION

Pattern classification is the enabling technology for speech recognition, image understanding, target recognition, and other important signal processing applications. A pattern classifier is a decision-making algorithm which determines the class label of a feature vector presented to the classifier. Based on statistical decision theory, artificial intelligence, fuzzy logic theory, and many other approaches, numerous types of pattern classifiers have been developed [3]. However, it remains an open question on which pattern classifier to use given a particular problem on hand. It is generally accepted that a universal pattern classifier which will outperform every other pattern classifiers is unlikely to be found.

A practice which is gaining popularity is to combine the output of several pattern classifiers, using, say the majority voting combination method. The situation is analogous to the decision making process in human society where many experts, each specialize

0-7803-4256-9/97/$10.00

in a sub-fields, are often summoned to form a **committee** to solve a complicate problem in a collective manner. The belief is that collective efforts can often arrive at a superior decision than by any individual expert. A pattern classifier formed by a combination of several member classifiers is called a **committee classifier** in this study. A number of prior studies of committee classifiers have been reported There are several empirical studies of combining multiple classifiers reported in literature [1], [2], [5], [6], [10], [11], [14], [15], [16], [4], [9], [12], [13], [17], [18]. We note there is another family of algorithms for combining multiple classifiers, called mixture of expert (MoE) approach [7], [8], [19]. In the MoE approach, the output of member classifiers are linearly combined with weights which are functions of input feature vectors through the use of a "gating network". In this study, we distinguish committee classifiers from Mixture of experts by restricting the combination rules of a committee classifier to be dependent on output of member classifiers only, and not directly dependent on the inputs to each member classifier.

Many of these works reported that the enhancement of classification rate of a committee classifier will be maximized when its member classifiers are independent to each other. This is often done assuming the committee classifier's output is an ensemble average (linear combination) of those of its member classifiers. Such an analysis is more suitable when the member classifier's output is interpreted as an estimate of the posterior probability of a given feature vector belongs to a specific class. For those classifiers whose output is binary valued, such an analysis is not quite applicable. In [5], the voting mechanism of member classifiers is analyzed assuming the output of each classifier obeys a binomial distribution. Some asymptotic behavior of such a majority committee classifier has been given. In general, majority voting is a simple, yet effective combination method.

Wolpert [15], [16] used the term "stack generalization" to describe a general committee classifier whose output may be multiple level nonlinear combination of lower level member classifiers. Voting can be regarded as a special case of stack generalizers. In this study, we will analyze what is the best performance a nonlinear committee classiifer can achieve given that each member classifier gives only binary output and is fixed (i.e. can not be modified or trained). We identified a phenomenum called "aliasing" which corresponds to the situation that feature vectors with different class labels having the

same output combination from all member classifiers. When an alias occurs, the combined classifier will never reach perfect classification.

In section II, some basic notions of a committee classifier will be reviewed briefly. In section III, the alias phenonum will be analyzed. Some simulation results will be reported as well. In the following discussion, we will denote $\mathbf{x}$ to be the current feature vector presented to a classifier, and $\mathbf{y}(i)$ as the output of the i^{th} expert classifier. The output of the combined committee classifier will be denoted by $\mathbf{z}$.

II. LINEAR COMMITTEE CLASSIFIER

Committee classifier consists of a committee of n individual pattern classifiers. There outputs, denoted by $\{\mathbf{y}(i);\ 1 \le i \le n\}$ are to be combined, linearly or non-linearly, via a set of combination rules, to form the final output, $\mathbf{z}$. For classification problem, the outputs $\mathbf{y}(i)$ and $\mathbf{z}$ are c by 1 vectors with a "1" in an k^{th} entry indicating the classifier decides that the input feature vector $\mathbf{x}$ belong to the k^{th} class. Usually, one would allow only one element to be 1 and the rest should remain at 0.

A linear committee classifier approach is based on the assumption that each classifier's output is a real number between 0 and 1 and can be interpreted as an estimate of the posterior probability of $\mathbf{x}$ is drawn from class i given its value $\mathbf{x}$, $P\{i|\mathbf{x}\}$. A model of the i^{th} classifier's output can be written as: $\mathbf{y}(\mathbf{x},i) = P\{i|\mathbf{x}\} + \varepsilon(i|\mathbf{x})$ where $\varepsilon(i|\mathbf{x})$ is a random estimation error with zero mean and variance $\sigma_i^2(\mathbf{x})$. Then the objective is to find a set of weights $\{w(i);\ 1 \le i \le n\}$ such that the variance of the overall linear estimate

$$\|\mathbf{z}(\mathbf{x}) - \sum_{i=1}^{n} \mathbf{y}(\mathbf{x},i)w(\mathbf{x},i)\|^2 \qquad (1)$$

is minimized. This *minimum variance estimate* so obtained can be found as:

$$w(\mathbf{x},i) = \frac{1}{\sigma_i^2(\mathbf{x}) \sum_{i \neq j} [1/\sigma_j^2(\mathbf{x})]}. \quad (2)$$

In other words, the weights are inversely proportional to the variance of the estimate. To apply this method, one must estimate the error variance of $\sigma_i^2(\mathbf{x})$ for each expert classifier.

III NONLINEAR COMBINATION RULES

Nonlinear combination rules can be regarded as a general meta-classifier designed to classify a concatenated feature vector $\mathbf{y}(\mathbf{x}) = [\mathbf{y}(\mathbf{x},1)\ \mathbf{y}(\mathbf{x},2) \bullet \bullet \bullet \mathbf{y}(\mathbf{x},n)]$. Any known classifier structures, such as MAP (maximum a posterior probability), kNN (k nearest neighbors), SOM (self-organization map), decision trees (e.g. ID3), can be applied to serve this purpose. The question is, is there any way to predict how the committee classifier performs compared to individual member classifiers?

x1	x2	x3	y1	y2	y3	y4	y(desired)
0	0	0	0	0	1	1	0
0	0	1	0	1	0	1	1
0	1	0	1	1	1	0	1
0	1	1	1	0	1	1	1
1	0	0	0	0	0	1	0
1	0	1	0	1	0	0	0
1	1	0	1	0	1	0	0
1	1	1	1	1	1	1	1

Figure 1. Examples illustrating aliasing effect of a committee classifier

Let us examine a special case illustrated in figure 1 when both inputs (feature vectors) and outputs of each classifier are discrete value in {0, 1}. In this case, there are only 2^k different input combinations where k is the feature vector dimension. In other words, we have a Boolean function realization problem on hand. Let us focus on a single class at a time. We now have a truth table similar to one shown in figure 1 (shadowed cells indicate misclassification of the corresponding

classifier). In this figure, y is the desired mapping and y1-y4 are 4 imperfect member classifiers. The question now is: Given y1, y2, y3, y4, can a meta-classifier (combination rules) be defined so that it gives an output which is the same as y? Here is an incomplete 4 Boolean variable minimization problem, and one of the solution is: y = y2•y4 + y1•y4 + y1•y2. In other words, combining y1, y2, and y4, the committee classifier is able to yield 100% classification rate on this training data set – a performance better than any individual classifier. On the other hand, if there are only y1, y2, y3 are available, note that when (y1,y2,y3) = 010, and 101, both of them appear twice with different values of y. Thus, one can choose only one of the values. This implies that the maximum classification rate will be at most 6/8 which is no better than either y1 or y2 alone. This phenomenon of having different target values associated to the same classifier output combination is called **alias**.

When $y(\mathbf{x},i) \in \{0, 1\}$, the feature space X is decomposed into disjointed regions labeled by the derived feature vectors (y1, y2, • • •). Alias can be regarded as the mis-classification errors within each of these regions. Therefore, an optimal combination rule would be to assign each of these regions to a class label which minimize the alias error. As a matter of fact, in this case, the optimal committee classifier amounts to a look-up table which maps from each binary vector to a specific class label.

IV. EXPERIMENTATION

We have employed the four data sets from machine learning database at UCI. They are: A. credit card applications, B. breast cancer diagnosis, C. DNA Promoter sequence recognition, and D. poisonous mushroom identification. Each data file is randomly partitioned into three parts. A *three-way cross-validation* procedure is adopted to better estimate the generalization error: Each method is applied three times (trials) to each data file. In each trial, two of the three parts are used as training data, and the third as the testing data. After three trials, each of the three parts of the original data file will be tested exactly once. The testing error rates of the three trials then are averaged to yield the overall classification rate of a particular classification method on a given data set.

Four expert classifiers are used in this experiment: a 3-nearest neighbor (3NN) classifier, a maximum likelihood (ML) classifier, a Learning vector quantization (LVQ 2.1) classifier, and a multi-layer perceptron (MLP) classifier. In developing the ML classifier, feature vectors in each class is assumed to have a normal distribution. Thus, it effects a linear classifier. With the MLP classifier, a 2-layer, fully connected configuration (one hidden layer) is used, with 10 hidden units – a number assigned arbitrarily. Since our objective here is not to compare performance of individual classifiers, sub-optimal implementation of these classifiers should not prevent us from comparing results between the committee classifier to the best of the individual classifiers. Each of these four classifiers will be used as the committee classifier classifying not only the output of the member classifiers, but also the original feature vector to facilitate aliases-free classification. All but the LVQ algorithm are implemented with Matlab (v.4.2c) m-files, tested on a HP workstation. The LVQ algorithm is implemented by the SOM research group of the University of Helsinki, and is available at *ftp://cochlea.hut.fi/ pub/.*

Note that for each data set and each classification method, there are actually three different expert classifiers developed – each developed on one of the three different training data set. Thus, all experiment are performed three times on these three different partitions, and the results are reported as the average of three trials. The distribution of these inputs are summarized in the following table 1 to table 3 below. The order of outputs are ML-3NN-LVQ.

Output of	Cancer 1		Cancer 2		Cancer 3	
Experts	Class 1	Class 2	Class 1	Class 2	Class 1	Class 2
111	109	2	110	1	106	1
112	0	0	1	0	0	3
121	0	6	0	2	0	5
211	0	0	0	0	0	0
222	0	11	0	6	0	8
221	0	0	0	0	0	0
212	0	0	0	0	0	0
122	0	46	4	50	2	49

Table 1. Cancer data set output.

Output of	Card 1		Card 2		Card 3	
Experts	Class 1	Class 2	Class 1	Class 2	Class 1	Class 2
111	60	6	69	8	58	8
112	0	0	2	0	1	0
121	3	2	2	0	3	0
211	12	5	7	5	14	12
222	6	57	11	55	7	48
221	5	4	2	1	3	6
212	5	6	5	3	7	5
122	1	0	1	1	0	0

Table 2. Card data set output.

Output of	Heart 1		Heart 2		Heart 3	
Experts	Class 1	Class 2	Class 1	Class 2	Class 1	Class 2
111	73	12	73	12	73	13
112	2	3	3	7	0	5
121	4	6	1	2	7	3
211	4	1	3	7	3	5
222	14	58	13	67	17	45
221	4	1	5	9	2	0
212	4	5	0	1	4	3
122	5	34	3	24	5	45

Table 3. Heart data set output.

The shaded area in these three tables have two or three outputs indicating class 2, and according to majority voting rule, it should be classified as class 2. We see that aliases do occur. For example, in table 3, under the column of Heart 1, while all three classifiers' outputs indicating class 1, there are 12 samples actually belong to class 2. Therefore, no matter how smart the committee machine will be, it will be unable to distinguish these 12 samples as class 2. This is verified in the following experiment: We construct an induced feature vector which consists of the outputs of each of the four classifiers. Then we develop a committee classifier to classify these extended feature vectors using each of the four types of classifiers (3-NN, ML,

LVQ, MLP). Again, three trials are performed on each of the three different partitions of each data set, and the results are reported below:

	Voting	ML	3-NN	LVQ 3.0	Optimal
Cancer	4.98%	2.49%	3.64%	4.02%	2.11%
Card	19.18%	19.18%	19.77%	19.18%	18.41%
Gene	21.98%	26.86%	22.66%	14.88%	13.79%
Heart	22.03%	22.09%	22.90%	22.03%	20.00%

Table 4. Classification Error rates of committee classifiers with outputs of member classifiers only

From this table, we observe that the committee combination will at times out-performs the majority voting significantly. However, some combinaton methods, notably the 3-NN methods consistently performs worse than the simple voting. This is because there are only 8 different distinct input data samples, and 3-NN is inadequate to perform classification when the number of distinct data samples are too few. The entries in the column labelled with "optimal" are minimum classification errors can be achieved by the committee classifier given the component classifiers. It is a lower bound. The errors incurred in this column are due entirely to the aliasing effect discussed earlier.

From above results, we observe that compared to simple majority voting, the committee machine approach, at least with this experiment, does not significantly improve the classification performance in general. Among the three different classifiers, LVQ 3.0 seems consistently out-perform the voting method, while other two classifiers gives mixed results. Compare the committee method, and the extended committee method, where original features are used, the results are mixed. Our preliminary explanation is that the additional dimension causes the ML or 3-NN based committee classifier confused.

REFERENCES

[1] Battiti, R., and A. M. Colla, "Democracy in neural nets: voting schemes for classification," *Neural Networks*, vol. 7, no. 4, pp. 691-709, 1994.

[2] Drucker, H., C. Cortes, L. Jackel, Y. LeCun, and V. Vapnik, "Boosting and other ensemble methods," *Neural Computation*, vol. 6, no. 6, pp. 1289-1301, 1994.
[3] Duda, R. O., and P. E. Hart, *Pattern classification and scene analysis*. New York: Wiley, 1973.
[4] Freund, Y., and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an appplication to boosting," in Proc. *Proc. 2nd European Conf. on Computational Learning Theory*, 1995,
[5] Hansen, L. K., and P. Salamon, "Neural network ensembles," *IEEE Trans. on PAMI*, vol. 12, no. 10, pp. 993-1001, 1990.
[6] Ho, T. K., J. J. Hull, and S. N. Srihari, "Decision combination in multiple classifier systems," *IEEE Trans. on PAMI*, vol. 16, no. 1, pp. 66-76, 1994.
[7] Jacobs, R. A., M. I. Jordan, S. Nowlan, and G. E. Hinton, "Adaptive mistures of local experts," *Neural Computation*, vol. 3, no. pp. 79-87, 1991.
[8] Jordan, M. I., and R. A. Jacobs, "Hierarchical mixtures of experts and the EM algorithm," *Neural Computation*, no. pp. 1993.
[9] Krogh, A., and J. Vedelsby, "Neural network ensembles, cross validation and active learning," in *Advnaces in Neural Information Processing Systems 7*, Ed(s)., Cambridge MA: MIT Press, 1995,
[10] Meir, R., "Bias, variance, and the combination of estiamtors: the case of least linear squares," in *Advances in Neural Information Processing Systems 7*, Ed(s)., Cambridge, MA: MIT Press, 1995,
[11] Perrone, M. P., and L. N. Cooper, "When networks disagree: ensemble method for neural netowrks," in *Neural Netowrks for Speech and Image Processing*, R. J. Mammone, Ed(s)., Chapman-Hall, 1993,
[12] Tresp, V., and M. Taniguchi, "Combining estimators using non-constant weighting functions," in *Advances in Neural Information Processing Systems 7*, Ed(s)., Cambridge MA: MIT Press, 1995,
[13] Tumer, K., and J. Ghosh, "Error correlation and error reduction in ensemble classifiers," *Connection Science, special issue on combining neural networks*, no. pp. (to appear), 1996.
[14] Twomey, J. M., and A. E. Smith, "Committee networks by resampling," in *Intelligent Engineering Systems Through*

Artificial Neural Networks, C. H. Dagli et al, Ed(s)., ASME Press, 1995, pp. 153-158.

[15] Wolpert, D. H., "Stacked Generalization," *Neural Networks*, vol. 5, no. 2, pp. 241-259, 1992.

[16] Wolpert, D. H., "Combining generalizers using partitions of the learning set," in *1992 Lectures in Complex Systems*, L. Nadel and D. Stein, Ed(s)., Addison Wesley, 1993, pp. 489-500.

[17] Wolpert, D. H., and W. G. Macready, "Combining stacking with bagging to improve a learning algorithm", Santa Fe Institute,Technical Report,Aug. 22, 1996.

[18] Xu, L., A. Krzyzak, and C.Y. Suen, "Methods of combining multiple classifiers and their applications to handwriting recognition," *IEEE Trans. on SMC*, vol. 22, no. pp. 418-435, 1992.

[19] Xu, L., and M. I. Jordan, "EM learning on a generalized finite mixture model for combining multiple classifiers," in Proc. *Proc. World Congress on Neural Networks*, Portland, OR, 1993,

UNSUPERVISED SPEAKER CLASSIFICATION USING SELF-ORGANIZING MAPS (SOM)

Itshak Voitovetsky, Hugo Guterman and Arnon Cohen
Department of Electrical and Computer Engineering
Ben-Gurion University of the Negev
P.O.B. 653, Beer-Sheva, 84105
Israel
Tel: +972-7-6472417
Fax: +972-7-6472949 (Attn. Itsik)
itsik@newton.bgu.ac.il

Abstract

An algorithm for unsupervised speaker classification using Kohonen SOM is presented. The system employs 6x10 SOM networks for each speaker and for non-speech segments. The algorithm was evaluated using high quality as well as telephone quality conversations between two speakers. Correct classification of more than 90% was demonstrated. High quality conversation between three speakers yielded 80% correct classification. The high quality speech required the use of 12^{th} order cepstral coefficients vector. In telephone quality speech, additional 12 features of the difference of the cepstrum were required.

INTRODUCTION

Speaker recognition (identification and verification) is being used in many commercial, military and forensic applications. Usually the problem is defined as supervised classification, where *a-priori* knowledge on the speakers is available so that pre-training can be performed [1-4]. In many applications, however, no such *a-priori* knowledge is available. Unsupervised methods must be used.

Solutions to various aspects of the problem have been suggested in the literature. The application of hierarchical NN was described in [5], and HMM based systems in [6-9]. Other methods based on EM algorithm for Gaussian mixture estimation [10], and various VQ methods [11-13], were also employed .

In general, given a multi-speaker conversation, the algorithm has to estimate the number of speakers, to segment the speech signal and to assign each segment to its speaker. The problem has been also termed "speech segmentation" [10-11]. In our current application the number of speakers, R, is assumed to be known. Generally, during a conversation, it may happen that one speaker interferes with another. We assume that the speech signal does not contain such interference, namely simultaneous speech does not occur. All segments with simultaneous

0-7803-4256-9/97/$10.00 ©1997 IEEE

speech are currently manually eliminated from the data prior to performance evaluation, (during the training process all the data was used).

We suggest here an unsupervised classification system that first makes a preliminary segmentation into speech/non-speech segments, using only "energy" threshold. The system then automatically trains R+1 Kohonen SOM [14]: R for the speakers and one for non-speech segments. Initial conditions are set, and then all neural networks (NN) compete among themselves until a balance is achieved.

There were four reasons why Kohonens' SOM was chosen. First, an unsupervised learning algorithm was required because of the problem definition. Second, due to short segments, multiple centroids are required to describe each speaker. Third, when we use SOM's, every SOM defines a different speaker model (or non-speech model). If we use one large network, it would be impossible, to indicate which centroids (or neurons) belongs to the same model. For this reason other unsupervised networks such as ART2 [15], or the network architecture proposed by Nissani [16], cannot be used. Fourth, every SOM is a trained code book (CB), this means that it can be used as CBs for discrete HMM that can later be used for (supervised) speaker recognition.

SYSTEM'S ARCHITECTURE

The general block diagram of the system is shown in figure 1.

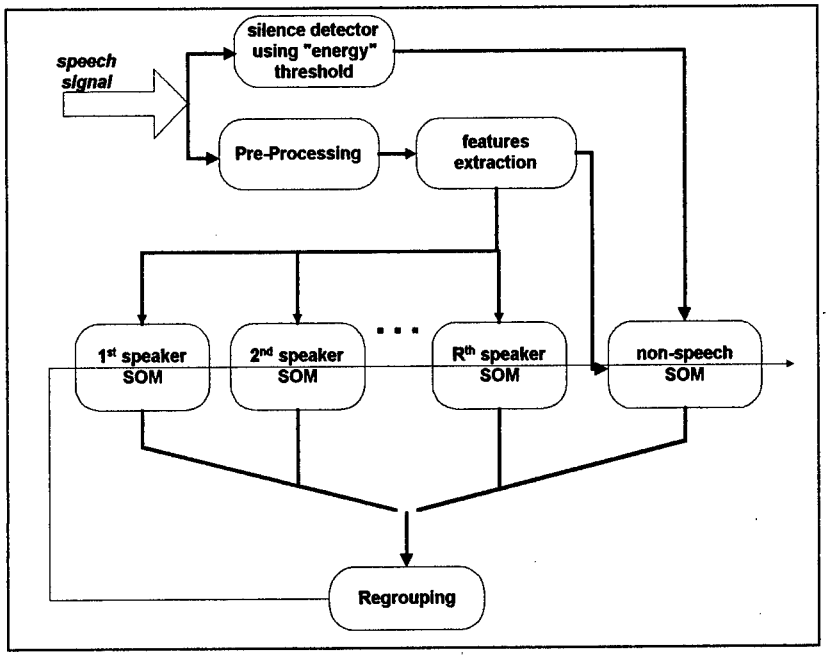


Fig. 1: General description of the unsupervised speaker classification system.

The speech analysis was based on overlapping 15 milliseconds analysis frames, with 5 millisecond frame rate. Each frame was represented by a features vector which included the 12^{th} order cepstral coefficients, estimated from the 12^{th} order LPC. In the telephone data, the features vector was augmented by the 12^{th} order first difference cepstral coefficients [1]. In addition the mean absolute value of an accumulated 50 millisecond frame was also calculated, for speech/non-speech evaluation.

Rough segmentation of speech and non-speech data was performed by thresholding the absolute value feature. The threshold level was set at three percent of the maximum, for high quality speech and one percent for telephone data. The levels were determined experimentally. The fact that higher level was required for high quality speech seems illogical. It is probably due to the fact that in the high quality data the variance of the speech amplitude is much lower than that of the telephone speech. The use of more sophisticated speech detection algorithms should explored hear.

The initial conditions to the system were determined as follows: all segments, classified by the rough speech/non-speech classifier as non-speech, were used to train the non-speech network. Segments roughly classified as speech segments were randomly and equally divided and used to train the R speaker models.

Each one of the models (including the non-speech model) was a Kohonen 6x10 SOM. Each SOM was trained by the Kohonen algorithm [14]. The inputs to the SOM were the cepstrum, or cepstrum and difference cepstral coefficients. The outputs of the SOM were Euclidean distances between input vector and network's weight vectors. In each iteration, at the end of the training process, regrouping process was employed. The grouping process was performed with a segments of 100 frames (0.5 second).

The algorithm is based on clustering the data in such a way that a total error criterion, during regrouping, is minimized.

Let $\mathrm{d}_{n,k}^{(m)}(r)$, be the Euclidean distance between the n-th-vector of the k-th segment ($v_{n,k}$) and the closest centroid in the r-th model, during iteration m:

$$\mathrm{d}_{n,k}^{(m)}(r) = \mathrm{d}\left(v_{n,k}, c^{(m)}(r)\right) = \left(v_{n,k} - c^{(m)}(r)\right)^T \left(v_{n,k} - c^{(m)}(r)\right) \qquad (1)$$

In the m-th iteration, the total distance between the k-th segment and the r-th model, $\mathrm{D}_k^{(m)}(r)$, is given in (2).

$$\mathrm{D}_k^{(m)}(r) = \sum_{n=1}^{100} \mathrm{d}_{n,k}^{(m)}(r) \qquad (2)$$

The k-th segment, S_k, is assigned to the model j (SOM_j) yielding minimum total error:

$$j = \arg\min_{r=0,\ldots,R}\left\{\mathrm{D}_k^{(m)}(r)\right\} \Rightarrow S_k \in SOM_j \qquad (3)$$

Hence an iteration of the process is defined by:

1. Retrain the models with the new clusters, achieved by the previous iteration.
2. Regroup the data using (3).
3. Check for termination: If termination criterion is met, exit, if not return to step 1.

It has been proved that this algorithm converges [17].

At the end of this iterative procedure, the system provides R+1 models, for the R speakers and for non-speech data. The data is segmented and labeled as required.

The termination criterion used here was based on the regrouping. Termination was declared when two consecutive iterations showed no change in the clusters. It is of course possible to use a less restrictive criterion which will require that two consecutive iterations will exhibit a change of no more than a given predetermined level. The use of such a criterion will reduce computation time at the expense of accuracy.

CLASSIFICATION ERROR EVALUATION

The algorithm is based on the classification of 0.5 second segments. Each segment may be assigned to one model (speaker or not-speech model) or, in transient segments, due to the finite resolution, may be common to two models or more. The definition of the classification error is clear in the non-transient segments. In case of transient segments, the correct assignment may be to either one of the correct models. Obviously, it makes sense to define classification error that takes in account a segment split between models. A linear piecewise classification error weight is used here.

Figure 2 shows 10 seconds (200 frames per second) of manually classified speech and the error weighting. The dashed lines show an example where a segment includes speech from both the first and the second speakers.

The error weighting has been defined as follows:

1. From the manual segmentation of the speech, all transient times, namely the switching times between speakers were found and denoted: $\{n_1, n_2, \ldots, n_M\}$.
2. In the neighborhood of every transient time a local error weighting function, was defined as:

$$w_{\mathrm{m}}(n) = \begin{cases} \left|\dfrac{n-n_m}{L/2}\right| & ; \quad |n-n_m| < \dfrac{L}{2} \\ 1 & ; \quad \text{otherwise} \end{cases} \qquad (4)$$

where L is the segment's duration (L=500msec. in our case).

3. Sum all the local weighting functions and subtract $(M-1)$:

$$g(n)=\sum_{m=1}^{M} w_{\mathrm{m}}(n)-(M-1) \tag{5}$$

4. The general weighting function will be:

$$w(n)=\begin{cases} g(n) & ; \quad g(n)>0 \\ 0 & ; \quad \text{otherwise} \end{cases} \tag{6}$$

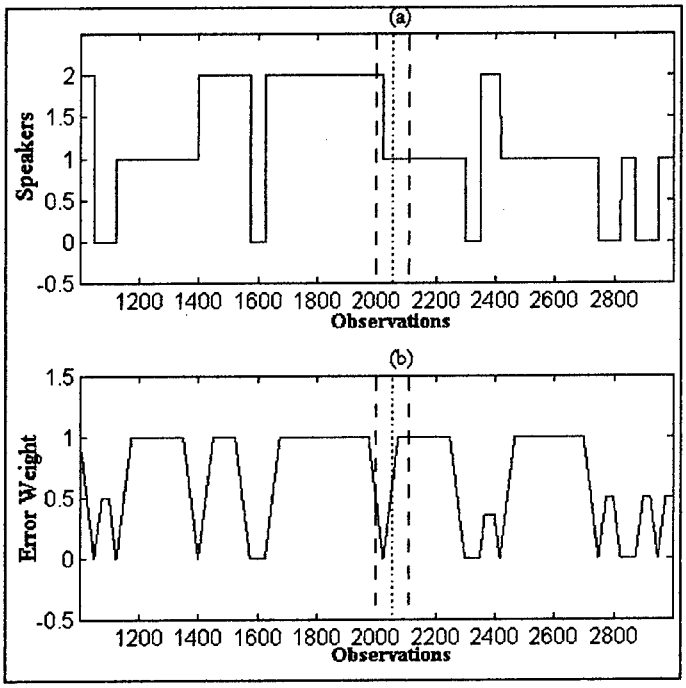


Fig 2: Error weighting function ("0"-non-speech, "1"- speaker A, "2"- speaker B). a) 10 seconds of manual segmentation, b) Weighted error function.

THE DATA BASE

The Hebrew data base, used for the evaluation of the system, consisted of 9 files with two speakers, 3 files with three speakers, all of high quality speech dialogue, and 12 telephone dialogues. The duration of the high quality speech files were 72-180 seconds per file. Telephone files duration were about two minutes per

file. The high quality speech (7.8khz bandwidth) was sampled in an acoustic room, at 16KHz sampling rate and 12Bit resolution. Five males and one female participated in the conversations. One of the speakers took part in all dialogues. The telephone quality dialogues were recorded from the telephone line. The data was filtered by a 3.8KHz low pass filter and sampled at 8KHz sampling rate, with 12Bit resolution. Twenty four male speakers participated in the dialogues.

RESULTS

The algorithm was evaluated with the small data base described above. All high quality conversations between two speakers, where tested on segments of half second, without overlapping. It was found that most of the errors between automatic and manual segmentation were due to transitional segments and the relatively poor resolution of the system. Table 1 shows a sample of the results for high quality speech.

TABLE 1: CONFUSION MATRIX (HIGH QUALITY SPEECH)

	weighted error		
	A	B	NS
A	93.5	0	1.6
B	4.3	94.9	3.4
NS	2.2	5.1	95.0
Total Error [%]	5.6		

For high quality speech, using the part of the data base with two males conversation, the error was between 5.5% and 6.0%. For male/female dialogue the error was only 4.3%.

The algorithm was also evaluated with, two speakers, telephone quality speech. When 12^{th} order cepstrum features were used and half a second segments without overlapping were employed, classification results were very poor (11-47% weighted error). Augmenting the features vector with 12 delta-cepstrum features and 75% overlapping - the results improved significantly. Eight (out of the total of 12) conversations significantly improved their classifications (approximately 6% weighted error). The confusion matrix is presented in table 2. Other 4 (out of the 12) did not converge. These four files were examined by a human listener. The files were found to be of very low quality. One of the files was judged by the listener as having three, rather than two speakers.

We have tried to apply a 3 speakers (plus non-speech) networks to these files. The non-speech segments were all well classified. Two of the four files converged into two separate clusters (with error of about 15%) and one extra cluster that contained segments of both speakers. The third file had one good cluster, one cluster containing segments of both speakers and one extra cluster that contained

simultaneous speech. The last file converged into two good clusters and one extra cluster that contained breath sounds, coughs and other interferences.

High quality, three speakers conversation files were processed with features and segmentation similar to the ones used in telephone quality speech. The results were not as good as for two speakers case (19.5% classification error).

TABLE 2: TELEPHONE CONVERSATION, CONFUSION MATRIX.

	weighted error		
	A	B	NS
A	93.6	2.4	1.8
B	0.9	94.0	4.6
NS	5.5	3.5	93.6
Total Error [%]	6.2		

Figure 3 shows a 10 seconds of segmented speech. It can be seen that except for very short segments of non-speech at the beginning and at 8 seconds, there is an agreement between the manual and automatic segmentation.

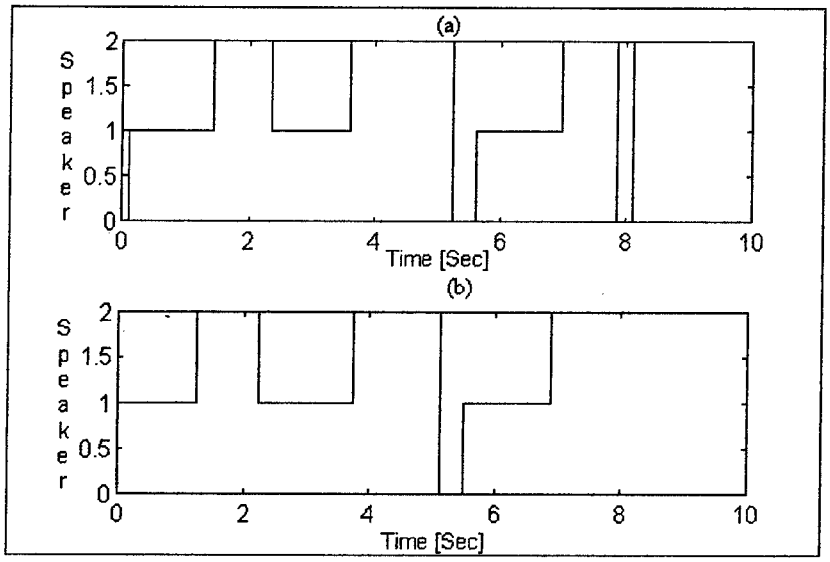


Fig. 3: 10 second classification of telephone conversation.
("0"-non-speech, "1"- speaker A, "2"- speaker B).
a) manual segmentation, b) SOM networks segmentation

Figure 4 shows an example of the system's convergence as a function of iteration number. In case of two speakers, thirty to forty iterations are needed for

convergence for one minute of telephone conversation. For two minutes of data , 50-65 iterations are usually needed. However, after 20 iterations, the system usually yields results close to optimal. In practice, about 20 iterations will be needed.

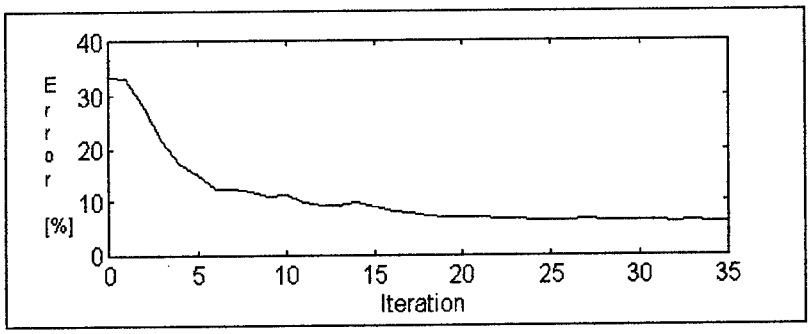


Fig. 4: The weighted error as a function of iteration number, An example for convergence determination.

CONCLUSIONS

A new architecture for unsupervised speaker classification was presented, using Kohonen SOM. With two speakers, and cepstral coefficients, both high quality and most telephone quality conversations, yielded classification errors of about 6%.

The same data base was used with an unsupervised algorithm based on HMM [9]. The results of the two algorithms are compatible. More work is required in order to determine, with sufficient statistical significance, whether one algorithm is more accurate than the other. The computation time difference between the two must also be further examined before a conclusive comparison can be made.

The algorithm achieves better results for conversations between male and female. This result is not surprising, because of the differences in the voice characteristics of the sexes.

For conversations with 3 speakers, classification error is about 20%. The errors appear between the speakers. The non-speech model yields about 5% error, similar to the two speaker case. Note that the data duration in two and three speakers experiments were approximately the same. More data may be needed in order to improve the three speakers results.

The current algorithm assumes that the number of speakers is known. We are currently in the process of developing a validity algorithm which will estimate the number of speakers participating in the conversation. In addition we are working on increasing the resolution of the algorithm.

A pre-processing algorithm will be developed to detect incidents of simultaneous speech, to allow automatic removal of such incidents.

REFERENCE

[1] F. K. Song and A. E. Rosenberg, "On the use of instantaneous and transitional spectral information in speaker recognition," **IEEE Trans. Acoust., Speech, Signal Processing**, vol. 36, no. 6, pp. 871-879, June 1988.

[2] A. Cohen and I. Froind, "On text independent speaker identification using a quadratic classifier with optimal features," **Speech Communication**, vol. 8, no. 1, pp. 35-44, March 1989.

[3] S. Furui, "Cepstral analysis technique for automatic speaker verification," **IEEE Trans. Acoust., Speech, Signal Processing**, vol. ASSP-29, no. 2, pp. 254-272, April 1981.

[4] L. Xu, J. Oglesby, and J. S. Mason, "The optimization of perceptually-based features for speaker identification," **ICASSP-89**, vol. 1, pp. 520-523, 1989.

[5] M. Zaki, A. Ghalwash, and A. A. Elkouny, "CNN: a speaker recognition system using a cascaded neural network," **International J. of Neural Systems**, vol. 7, no. 2, pp. 203-212, May, 1996.

[6] L. Wilcox, F. Chen, D. Kimber, and V. Balasubramanian, "Segmentation of speech using speaker identification," **ICASSP-94**, vol. 1, pp. 161-164, 1994.

[7] J. O. Olsen, "Separation of speakers in audio data," **Proc. of 4th European Conference on Speech Communication and Technology**, vol. 1, pp. 355-358, September, 1995.

[8] A. Cohen, and V. Lapidus, "Unsupervised speaker segmentation in telephone conversation," **The Nineteenth Convention of Electrical and Electronics Engineers in Israel**, pp. 102-105, 1996.

[9] A. Cohen, and V. Lapidus, "Unsupervised, text independent, speaker classification," **ICSPAT-96**, pp. 1745-1749, 1996.

[10] M. -H. Siu, G. Yu, and H. Gish, "An unsupervised, sequential learning algorithm for the segmentation of speech waveform with multiple speakers," **ICASSP-92**, vol. 2, pp. 189-192, 1992.

[11] M. Sugiyama, J. Murakami, and H. Watanabe, "Speech segmentation and clustering based on speaker features," **ICASSP-93**, vol. 2, pp. 395-398, 1993.

[12] T. Kosaka, S. Matsunaga, and S. Sagayama, "Speaker-independent speech recognition based on tree-structured speaker clustering," **Computer Speech and Language**, vol. 10, no. 1, pp. 55-74, January, 1996.

[13] A. Cohen, and V. Lapidus, "Unsupervised text independent speaker classification," **The Eighteenth Convention of Electrical and Electronics Engineers in Israel**, pp. 3.2.2 1-5, 1995.

[14] T. K. Kohonen, "The self-organizing map," **Proc. IEEE**, vol. 78, no. 9, pp. 1464-1480, September, 1990.

[15] G. A. Carpenter and S. Grossberg, "ART 2: self-organization of stable category recognition codes for analog input patterns," **Applied Optics**, vol. 26, no. 23, pp. 4919-4930, December 1987.

[16] D. N. Nissani, "An unsupervised hyperspheric multi-layer feedforward neural network model," **Biol. Cybern.**, vol. 65, no. 6, pp. 441-450, 1991.

[17] I. Voitovetsky, H. Guterman, and A. Cohen, "Training algorithm convergence of multiple code book system," **Internal Report**, Ben-Gurion University of the Negev, Dep. of Electrical and Computer Engineering, 1997.

A SELF-SCALING NEURAL HARDWARE STRUCTURE THAT REDUCES THE EFFECT OF SOME IMPLEMENTATION ERRORS

H. Djahanshahi, M. Ahmadi, G.A. Jullien and W.C. Miller[1]

Department of Electrical Engineering, University of Windsor,
401 Sunset Avenue, Windsor, Ontario N9B 3P4, Canada
Email: djahans@uwindsor.ca

ABSTRACT

This paper explores a neural network hardware structure with distributed neurons that exhibits useful properties of self-scaling and averaging. In conventional sigmoidal neural networks with lumped neurons, the effects of weight errors and mismatches become more noticeable at the output as the network becomes larger. It is shown here that based on a stochastic model the inherent scaling property of a distributed neuron structure controls the output noise (error) to signal ratio as the number of inputs to an Adaline increases. Moreover, the averaging effect of distributed elements minimizes characteristic variations among neurons. These properties altogether provides a robust hybrid hardware with digital synaptic weights and analog neurons. A VLSI realization and an application of this neural structure are explained.

1. INTRODUCTION

One of the problems in the hardware implementation of a neural network is output error caused by various non-ideal elements. Analog neural network circuits are generally area-efficient but inaccurate, i.e. they are prone to problems such as gain errors, mismatches, offsets and drifts. In digital circuits, on the other hand, the main source of error is finite word length which in the case of synaptic elements is referred to as weight quantization effect. In order to realize dense and high-speed neural networks with large number of neurons for real world applications, the use of simple synapses and neurons with low precision weights and other types of non-idealities is unavoidable. The effect of implementation errors especially becomes more noticeable at the output when the network becomes larger [1].

In this paper we study a hybrid analog-digital neural network hardware with distributed neuron structure. Here, we are only concerned about errors and

1 The authors would like to acknowledge financial support from Natural Sciences and Engineering Research Council of Canada (NSERC) and the Micronet Network of Centres of Excellence. The authors also thank Canadian Microelectronics Corporation (CMC) for their VLSI design software, equipment and fabrication services.

quantization effects in recall hardware or, in other words, the implementation of an ideally-trained network.

Sensitivity to weight errors of neural networks with increasing number of neurons is analyzed in [1]. A stochastic model is developed to study an ensemble of networks with differing weights and the focus is on implementation of recall phase. We modify this model to study the properties of a *distributed* neuron hardware structure. It will be shown here, based on the modified model, that the self-scaling property of a distributed neuron controls its stochastic gain and hence reduces the output noise (error) to signal ratio when the size of the network grows. Also, the averaging effect of distributed elements minimizes mismatches across a sizable chip. A VLSI realization and an application of this hardware structure are also presented along with some simulation and experimental test results.

2. IMPLEMENTATION OF DISTRIBUTED NEURON

Figure 1 shows an Adaline with distributed neurons. In general, two main tasks of a neuron are summation of synaptic inputs, and nonlinear saturating function. If an Adaline is built with transconductance synapses and nonlinear resistive neurons, then summation is performed simply by hardwiring the outputs of synapses together. As shown in Fig. 1, a neuron of this type can be distributed into parallel elements having the same equivalent characteristic. In this case, equivalent resistive neuron receives the summation current and delivers, on the same supernode, an output voltage nonlinearly proportional to total synaptic input.

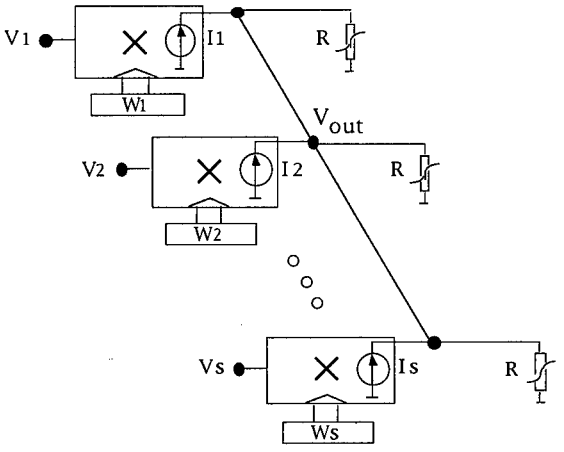


Fig. 1 Hybrid structure with distributed neuron

Each element of a distributed neuron can be integrated with one synapse to form a unified synapse-neuron (USN) block. A reconfigurable network based on distributed neurons is discussed in [2]. A hybrid analog-digital USN is presented by the authors in [3]. Here, experimental test results are presented from a recent CMOS fabrication. Fig. 2-a shows transistor-level diagram of a distributed neuron. Each element of neuron receives an average of input synaptic currents

(I_{ave}) and performs a nonlinear saturating I-to-V function by combining the quadratic characteristics of four MOS transistors.

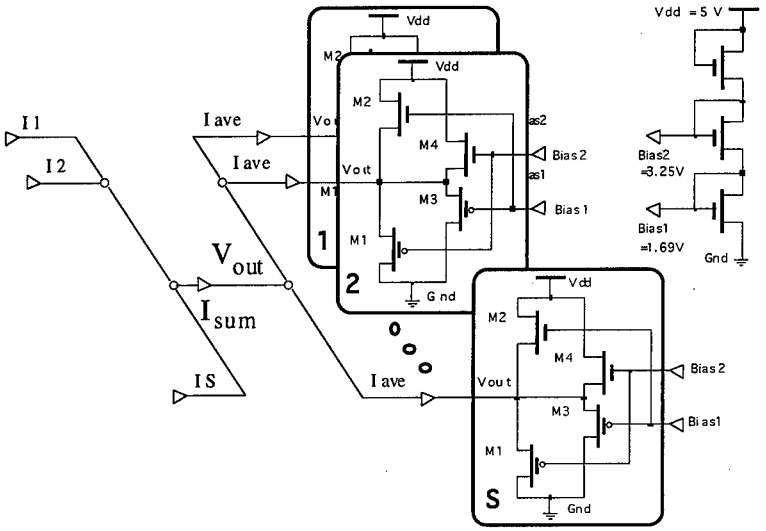


Fig. 2 a Implementation of distributed neuron

2.1. Self-scaling Property

Fig. 2-b shows the measured characteristics of 2-input and 5-input distributed neurons fabricated in 1.2μ CMOS. Distributed neuron structure exhibits an interesting self-scaling property. As the number of synaptic inputs (i.e. the number of neurons in previous layer) increases, the overall nonlinear characteristic stretches by itself. This property restores information received from extra inputs that would have been lost otherwise in large saturation areas of a fixed lumped neuron with increasing number of inputs.

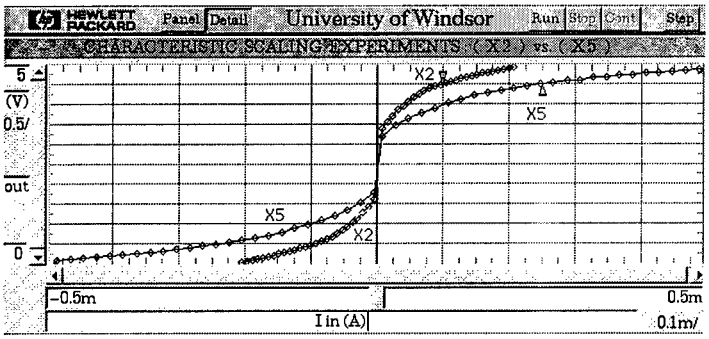


Fig. 2 b Experimental test results and self-scaling property

In fact, different applications require different number of neurons and neuron inputs. When the number of inputs to a lumped neuron increases, over-saturation

occurs. One method to circumvent this situation is to reduce synaptic activity via scaling down each weight by an arbitrary factor. This method is handy in software implementation. Equivalently, we should be able to use the same net input combined with a scaled activation function. Distributed neuron structure presents such scaling scheme. In fact, if the number of inputs to a distributed neuron is increased by a factor S, neuron will consist of S similar nonlinear resistive blocks in parallel (each possibly consisting of N original sub-blocks; here we assume N=1). As current divides equally among S similar blocks (each block receiving an average current I_{ave}) output voltage can be obtained in two alternative ways. If we consider the overall nonlinear function $F(.)$ we have, $V_{out} = F(I_{sum})$. On the other hand, regarding each individual block with nonlinear resistive function $f(.)$ we can write: $V_{out} = f(I_{ave.}) = f(\frac{I_{sum}}{S}) = f(\frac{\sum w_{ik}.V_k}{S}) = f(\sum \frac{w_{ik}}{S}.V_k)$. Since the two voltages must be the same, we conclude:

$$F(I_{sum}) = f(\sum \frac{w_{ik}}{S}.V_k) \qquad (1)$$

Therefore, a distributed neuron exhibits a self-scaling property which is equivalent to scaling down all the weights proportional to increase in the number of inputs. This property will be used in Section 3 to define a stochastic model for distributed neuron and to quantify the improvement obtained from this structure.

2.2. Averaging Effect

Analog lumped neurons implemented at different locations across a sizable chip are subject to noticeable variations in their expected characteristics. To demonstrate a worst case scenario, 2-input neurons with the same circuits as explained before are laid out as *lumped* cells at various locations, including corner positions, on a test chip. Measurements are performed on different cells and repeated over five fabricated chips. The worst case *on-chip* variations of the characteristic is found between two corner cells as 66mV in 5 Volt range (i.e. analog accuracy of 1.3%, approx. equivalent to 1 of 6 bits resolution). In Fig. 2-c a typical measured characteristic is shown on the left and a close-up of the worst case curves around 5V is shown on the right. The existing variations are related to fabrication process parameters, mainly the gradient of threshold voltage, VT, across the silicon die.

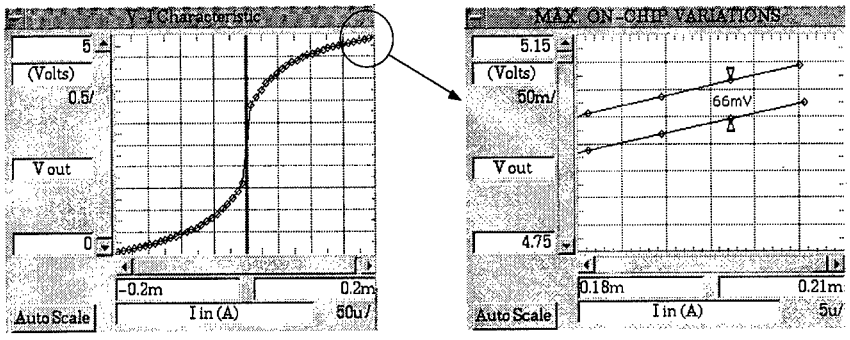


Fig. 2 c On-chip variations of characteristic (worst case)

The advantage of a truly distributed neuron can be observed when the building elements are distributed in one or two dimensions across the chip. In this case, an average of various characteristics is obtained which corresponds to average process parameters. The characteristic variations between two averaged neurons built in this manner is reduced only to the variability of two adjacent cells. In the case of our test chip, this mismatch was 26 mV in 5V range or 0.5%; as opposed to 1.3% for the case of lumped corner cells.

3. STOCHASTIC MODEL FOR DISTRIBUTED NEURON

For a conventional Madaline with lumped neurons, a stochastic model defines the ideal output of node n in layer l and the corresponding output error as follows [1]:

$$\mathbf{Y}_{l,n} = f(\mathbf{X}_l{}^T.\mathbf{W}_{l,n}) \qquad (\ \mathbf{X}_l{}^T \text{ stands for transposed matrix}) \tag{2}$$

$$\Delta\mathbf{Y}_{l,n} = f((\mathbf{X}_l + \Delta\mathbf{X}_l)^T.(\mathbf{W}_{l,n} + \Delta\mathbf{W}_{l,n})) - f(\mathbf{X}_l{}^T.\mathbf{W}_{l,n}) \tag{3}$$

$\mathbf{W}_{l,n}$, $\mathbf{X}_l$, $\Delta\mathbf{W}_{l,n}$ and $\Delta\mathbf{X}_l$ are independent identically distributed (iid) random vectors representing weights, inputs, weight errors and input errors respectively.

Based on this model, output Noise-to-Signal Ratio of layer l , defined as the ratio of the variance of the output error of layer l to the variance of the ideal output of layer l , is formulated as follows:

$$NSR_l = \frac{\sigma^2{}_{\Delta y_l}}{\sigma^2{}_{y_l}} = g(\sqrt{N}\sigma_x\sigma_w)\times(\frac{\sigma^2{}_{\Delta x}}{\sigma^2{}_x} + \frac{\sigma^2{}_{\Delta w}}{\sigma^2{}_w}) \tag{4}$$

The output *NSR* of a sigmoidal Adaline is expressed as a linear combination of input *NSR*, $\sigma^2{}_{\Delta x} / \sigma^2{}_x$ and weight NSR, $\sigma^2{}_{\Delta w} / \sigma^2{}_w$, and is amplified by a stochastic gain function $g > 1$. Gain g is an increasing function of its argument, $\sqrt{N}\sigma_x\sigma_w$, where N is the number of inputs to Adaline, and σ_x and σ_w are standard deviations of input and weight, respectively.

Thus, in a conventional neural network with lumped sigmoidal neurons an increase in the number of inputs, causes an increase in the stochastic gain, *g*, and hence an unwanted increase in output *NSR*. If the number of inputs to the Adaline increases by a factor *S* and the input and weight variances do not change, in the absence of any scaling scheme output *NSR* will increase as follows:

$$NSR = g(\sqrt{N.S}\sigma_x\sigma_w)\times(\frac{\sigma^2{}_{\Delta x}}{\sigma^2{}_x} + \frac{\sigma^2{}_{\Delta w}}{\sigma^2{}_w}) \tag{5}$$

In a distributed neuron structure, on the other hand, the characteristic of a neuron is changed adaptively based on the number of inputs. This self-scaling effect is a natural way of controlling g and hence decreasing *NSR*.

It has been shown earlier in Eq. (1) that self-scaling property of a distributed neuron is, in effect, equivalent to scaling down all the weights to the same saturating function while the number of inputs increases. Now, let us investigate the effect of this on output Noise (error) to Signal Ratio of an Adaline. If $w_S = w/S$ is defined as scaled weight, then we have $\sigma^2{}_{w_S} = \sigma^2{}_w / S^2$ and $\sigma^2{}_{\Delta w_S} = \sigma^2{}_{\Delta w} / S^2$. Thus,

$$NSR = g(\sqrt{N.S}\sigma_x\sigma_{w_S}) \times (\frac{\sigma^2{}_{\Delta x}}{\sigma^2{}_x} + \frac{\sigma^2{}_{\Delta w_S}}{\sigma^2{}_{w_S}}) = g(\frac{\sqrt{N}\sigma_x\sigma_w}{\sqrt{S}}) \times (\frac{\sigma^2{}_{\Delta x}}{\sigma^2{}_x} + \frac{\sigma^2{}_{\Delta w}}{\sigma^2{}_w}) \quad (6)$$

The terms in linear combination remain unchanged, while the gain factor is reduced due to the scaling of its argument by $1/\sqrt{S}$. The corresponding stochastic model is shown in Fig. 3. This property reduces *NSR* and improves the performance of recall hardware especially in large networks. The improvement is shown here by an example.

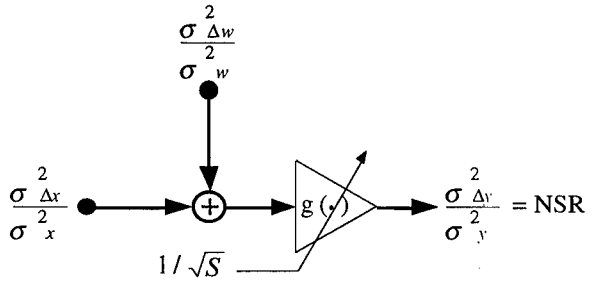


Fig. 3 Stochastic model of an Adaline with distributed neuron

Example: Suppose an Adaline with $N = 25$ inputs, for which inputs and weights are uniformly distributed over the range $[a,b] = [-2,2]$; therefore $\sigma^2{}_x = \sigma^2{}_w = (b-a)^2/12 = 4/3$. For an 8-bit quantization scheme, weights are quantized to levels equally spaced by $q = 1/64$; thus, weight error variance will be $\sigma^2{}_{\Delta w} = q^2/12 \approx 2\times10^{-5}$. Further, we assume $\sigma^2{}_{\Delta x} = \sigma^2{}_{\Delta w}$. For $\sqrt{N}\sigma_x\sigma_w > 2$, gain function defined in [1] may be approximated as: $g(\sqrt{N}\sigma_x\sigma_w) \approx 0.5 + 0.53\sqrt{N}\sigma_x\sigma_w$. Output noise-to-signal ratio of this Adaline is: $NSR = g(\sqrt{N}\sigma_x\sigma_w).(\frac{\sigma^2{}_{\Delta x}}{\sigma^2{}_x} + \frac{\sigma^2{}_{\Delta w}}{\sigma^2{}_w}) = g(6.67).(3\times10^{-5}) \approx 1.2\times10^{-4} \approx -39dB.$

Now if the number of inputs are increased to 100 (an increase by factor *S*=4), for a conventional lumped neuron characteristic, *NSR* is found from (5) as: $NSR = g(13.3).(3\times10^{-5}) \approx 2.3\times10^{-4} \approx -36.4dB$. If, instead, we use a distributed neuron structure, then from (6) we will obtain: $NSR \approx g(3.33).(3\times10^{-5})$ $\approx 6.8\times10^{-5} \approx -41.7dB$. In this example, *NSR* is reduced almost by a factor of 3, i.e. more than 5dB improvement. The effect would be even more noticeable for larger scaling factors.

4. VLSI APPLICATION

Figure 4 shows the schematic diagram of a 4-3-2 hybrid VLSI neural network with distributed neuron structure that is designed and implemented in 1.2μ CMOS using *Cadence* tools. This network is built with 23 similar blocks. Eighteen of these blocks are unified synapse-neurons (USN) each consisting of a Multiplying DAC synapse with 5-bit sign-magnitude weight and a portion of a distributed neuron. Five remaining blocks are used for neuron bias (threshold) adjustment. These units are in fact the same USNs on silicon with their nonlinear load disactivated.

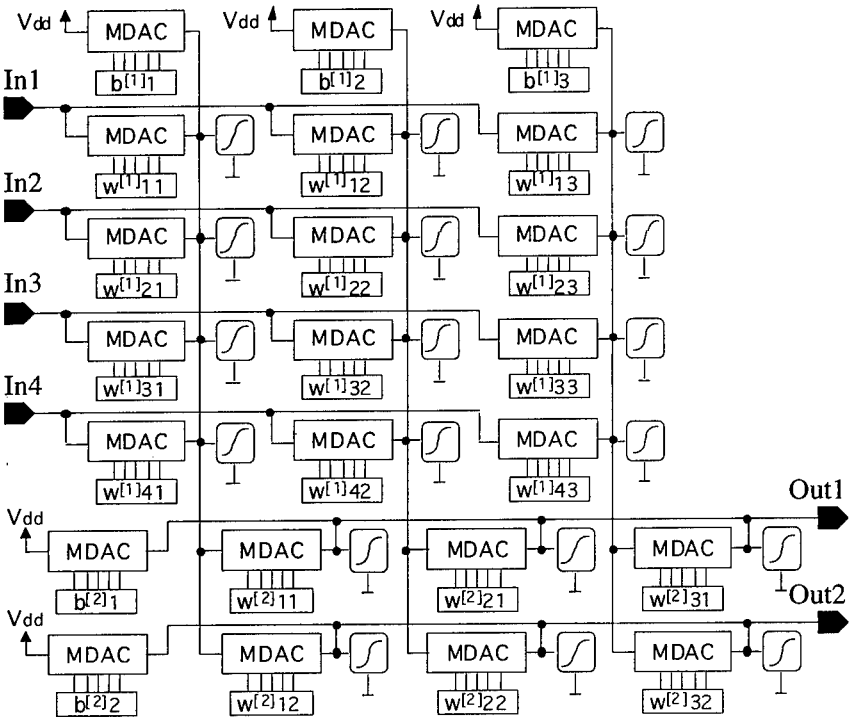


Fig. 4 Schematic diagram of VLSI implementation

This network is trained off-line for a 4-input template matching problem. An interactive Back-Propagation simulator based on *XView* programming on Sun-Sparc is used that allows user to define network architecture and I/O patterns. The resulting weights are rounded off to the resolution of hardware (5 bits) and a simulated recall is followed. When this final phase is passed, the weights will be programmed on chip. Weights (w's) and bias values (b's) for the above circuit are:

$$\mathbf{W}^{[1]} = \begin{bmatrix} -5 & 0 & -5 \\ 6 & 9 & 8 \\ 12 & -10 & 1 \\ -5 & 0 & -5 \end{bmatrix}, \quad \mathbf{W}^{[2]} = \begin{bmatrix} -14 & 6 \\ 5 & -15 \\ -7 & -8 \end{bmatrix}, \quad b^{[l]}{}_n = 4 \ \ (\forall l,n)$$

Figure 5 a shows four templates earlier introduced to the network during training. Outputs 1 and 2 respond to templates 1 and 2, respectively. Both outputs remain zero for templates 0 and 3. A complete transistor-level simulation is performed on 693 active and about 900 parasitic elements of recall circuit shown in Fig. 4. Figure 5 b shows typical post-layout results at 1 MHz input vector rate. To calculate each output vector, hybrid neural network performs 18 multiplications, 5 additions and 5 distributed nonlinear operations; an equivalent calculation rate of 28×10^6 /Sec. Current and power consumption on Vdd=5V are: $I_{ave.} \approx 730\mu A$ and $P_{ave.} \approx 3.65mW$. The circuit is functional at higher speeds *or* on lower supply voltages. For example, on supply voltage Vdd=3.3V, specifications are $I_{ave.} \approx 155\mu A$ and $P_{ave.} \approx 0.5mW$, i.e. 86% power saving compared to consumption on Vdd= 5Volts.

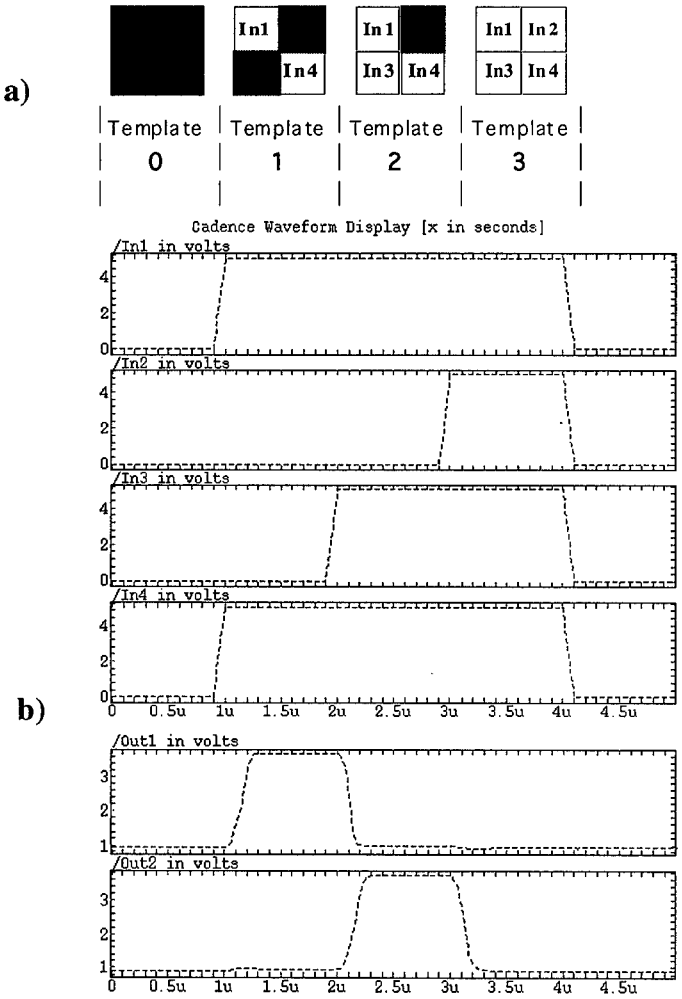


Fig. 5 Template matching problem and results

A chip is designed in 1.2μ CMOS (see Fig. 6) that contains two different versions of this network: **1)** A weight-programmable network with electronic inputs, **2)** A network with photosensitive inputs and pre-programmed weights for an optical template matching application.

With programmable synaptic weights network can be programmed to recognize different templates to be detected in different applications. The particular application that we are interested in is feature extraction in a handwritten numeral recognition system. This system consists of three stages: preprocessing, feature extraction and classification. Directional border codes are the features to be extracted by this network [4]. The basic process performed on a typical handwritten number and the 2x2 directional templates for feature extraction are illustrated in Fig. 7. More details about this system can be found in [5].

The same VLSI building blocks are conveniently used in the design of larger neural networks (e.g. [6]). In a network with 16 inputs, the required weight resolution for recall is still five (1 sign and 4 magnitude) bits. If lumped neuron blocks were used in a larger network; however, the quantization noise and mismatches would increase the output noise to signal ratio and in order to control *NSR* and avoid misclassification one has to: **a)** increase the weight resolution and analog circuit accuracies, or **b)** redesign new neuron blocks to effectively reduce their gain factor against implementation errors.

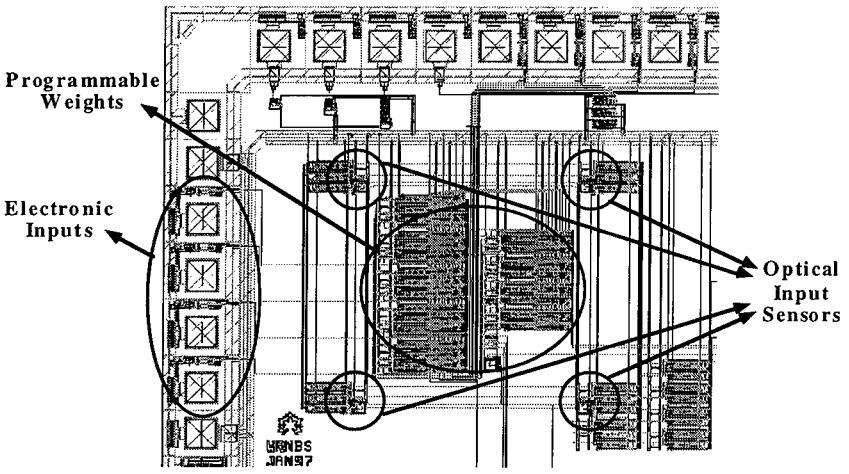


Fig. 6 Fabricated CMOS test chip

5. CONCLUSION

A robust neural network hardware structure with analog distributed neurons, digital weights and multiplying DAC synapses is presented. Two properties of this structure, namely self-scaling and averaging of neurons, are emphasized that both reduce the effect of some implementation errors. Experimental test results are presented for building blocks and a VLSI application is described.

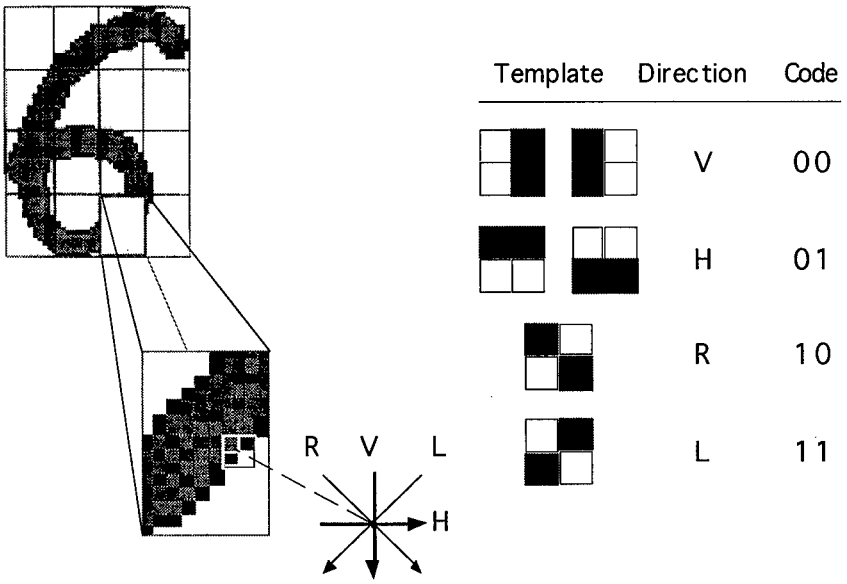


Fig. 7 Border feature extraction and directional templates

REFERENCES

[1] S. W. Piché, "The Selection of Weight Accuracies for Madalines", IEEE Transactions on Neural Networks, Vol. 6, No. 2, March 1995, pp. 432-445.

[2] S. Satyanarayana, Y. Tsividis and H.P. Graf, "A Reconfigurable VLSI Neural Network", IEEE Journal of Solid-State Circuits, Vol. 27, No. 1, January 1992, pp. 67 -81.

[3] H. Djahanshahi, M. Ahmadi, G.A. Jullien and W.C. Miller, "A Unified Synapse-Neuron Building Block for Hybrid VLSI Neural Networks", Proceedings of IEEE International Symposium on Circuits and Systems: ISCAS'96, Vol. 3, pp. 483-486, Atlanta, Georgia, May 12-15 1996

[4] J. Cao, M. Shridhar, M. Ahmadi and G.A. Jullien, "VLSI Implementation for Real-Time Extraction of Direction Vectors from Binary Images ...", Proceedings of 36th MWSCAS, Vol. 2, pp. 963-966, Detroit, August 1993.

[5] J. Cao, M. Shridhar, M. Ahmadi and G.A. Jullien, "Recognition of Handwritten Numerals with Multiple Feature and Multistage Classifier", Journal of Pattern Recognition, Vol. 28, No. 2, pp. 153-163, 1995.

[6] H. Djahanshahi, G.A. Jullien, W.C. Miller and M. Ahmadi, "Neural-based Smart CMOS Sensors for On-Line Pattern Classification Applications", Proceedings of IEEE International Symposium on Circuits and Systems: ISCAS'96, Vol. 4, pp. 384-387, Atlanta, Georgia, May 12-15 1996.

APPLICATION OF NEURAL NETWORKS TO THE PROBLEM OF FORECASTING THE FLOW OF THE RIVER NILE

Amir Atiya (*), Suzan El-Shoura (†),
Samir Shaheen (**), and Mohamed El-Sherif (†)

(*) Dept of Electrical Engineering, Caltech, MS 136-93,
Pasadena, CA 91125
(†) Electronic Research Institute, Tahrir Street,
Dokki, Giza, Egypt
(**) Dept of Computer Engineering,
Cairo University, Giza, Egypt

ABSTRACT:

This paper applies multilayer neural networks to the problem of forecasting the flow of the River Nile in Egypt. Estimating the flow of the River Nile can have significant economic impact, since it can help in managing scarce irrigation water. The second goal of the paper is utilize the time series as a benchmark to compare between different neural network forecasting methods. We compare between four different methods for input and output preprocessing, including a novel method proposed here based on the Discrete Fourier Series. We also consider the problem of forecasting several steps ahead. We compare between three methods for the multistep ahead prediction problem.

INTRODUCTION:

Neural networks have been used in many forecasting applications, for example stock market, exchange rate [1], and electric load forecasting [2]. We present here yet another forecasting application, namely river flow forecasting. Forecasting river flows is

an important application, that attracted the interest of scientists since more than 50 years. It can help in predicting agricultural water supply, predicting potential flood damage, estimating loads on bridges, etc.

In this paper the first goal is to apply neural networks to the problem of predicting the flow of the River Nile in Egypt. In addition, we exploit the river forecast problem as a benchmark to compare between different neural network forecasting approaches. The approaches, which all use multilayer networks with backpropagation training, are mainly different approaches to preprocess the time series and different approaches to solve the multistep ahead forecast problem.

ON THE FLOW FORECAST PROBLEM:

The river flow forecasting problem has been traditionally tackled using linear techniques, such as AR, ARMAX, and Kalman filter, and also using nonlinear regression (see [3]-[6]). We have found only one method in literature that uses neural networks, namely for the Huron River in Michigan [6]. Most of the forecasting methods consider one-day ahead forecast. For the River Nile a longer term forecast is more of interest, though it is more difficult than the one-day ahead problem.

The problem of flow forecast for the River Nile is particularly important for Egypt. Egypt depends almost exclusively on the Nile for agricultural irrigation. The flow of the River Nile exhibits a seasonal behavior. The flow is low during the winter months, and peaks during the months of August and September. The High Dam of Aswan (located South of Egypt) retains incoming water, and releases it in a more uniform way, so as to optimally fill agricultural and electricity generating needs (acting like the capacitor or the reservoir effect). Forecasting the flow of the river Nile can help in determining the optimum amount of water to release, and thus can help to more efficiently manage the water.

NEURAL NETWORK APPLICATION:

We used readings of the average daily flow volume for each ten-day period at the Dongola station, located in Northern Sudan. The readings spanned the period from 1974 to 1992. We used a network consisting of one hidden layer, with 3 hidden nodes in the hidden layer, and trained it using the standard backpropagation method. The network is trained for 4000 iterations. We used 150 points for each of the training stage and the testing stage. We used as error measurement the RMS normalized error, defined as the square root of the sum of squared errors, divided by the square root of the sum of squared desired outputs. Henceforth, when mentioning the "error", we will be meaning the RMS normalized error. As for the crucial design part of determining what inputs to use for the neural network, we have experimented with the following inputs:

1) the flows at the previous few time periods,
2) the flow at the same time period one year ago and two years ago,
3) the average of the flow of the last 12 months,
4) the period number (scaled by the number of periods), e.g. for the month of May the input would be 5/12.

We have performed simulations, and found the following observations:

1) There is a strong correlation between the training error and the testing error. This means that there is good generalization. Choosing a network/input set that gives a low training error will almost surely result in a low testing error.

2) In several exploratory runs we have found that no validation set was needed to determine optimal stopping point in training. The error for the test set goes down uniformly with iteration and does not bottom out.

3) For the majority of input combinations the results were somewhat similar. There were several cases which gave higher errors, but these were mostly for cases with insufficient number of inputs, and one can eliminate them easily

by observing the training error.

4) The forecasts for all periods of the year were quite accurate. Only for the peak flow periods there was some small error (see Figure 1 for the test forecast results of a ten-day ahead case). This suggests possibly training a separate network for the high flow periods, and using some kind of gating-type network (see [7], [8], [9]).

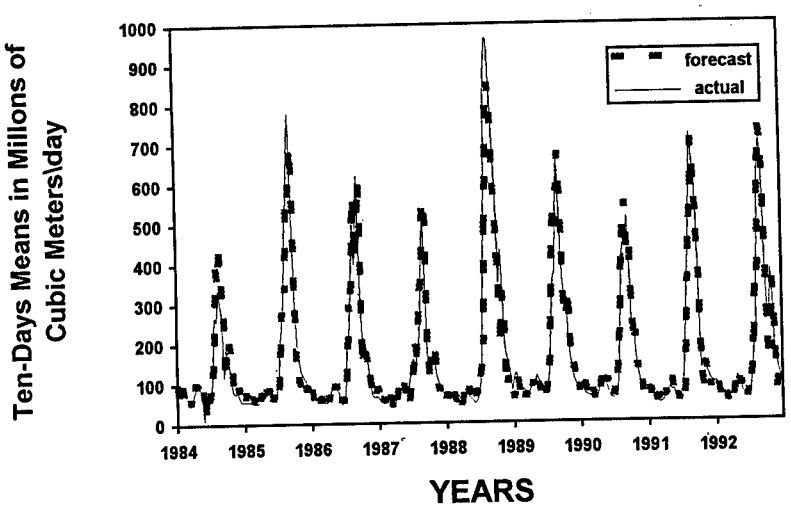


Figure 1: The results of the ten-day ahead forecast for the test period

ALGORITHM COMPARISON:

In addition to this basic neural network implementation, we used the data as a benchmark to compare between different neural network forecasting methods. The methods we used are the followoing:

Method 1) The neural network is trained to forecast the actual flow of the next time period.

Method 2) The neural network is trained to forecast the difference in flow between next period's flow and current period's flow (the desired output for the neural network at time t is $x(t+1) - x(t)$, properly scaled).

Method 3) We subtract the seasonal average of the flow to create a seasonally adjusted time series. Then we apply the neural network to forecast this seasonally adjusted series. This approach, hopefully, makes it a simpler problem for the neural network, by letting the neural network concentrate on forecasting the deviations from the seasonal average, rather than estimating both, the seasonal average and the deviation.

Method 4) A novel algorithm we propose here, that might be particularly suitable for seasonal (cyclical) time series. Let $x(t)$ be the time series values, and let the period be fixed (say T), but however the time series varies from cycle to the other, and can have a longer term variation or trend. Because of the seasonality of the time series, the Fourier series will be a natural representation of such series, and will carry much useful information relevant to the task of forecasting the series. Therefore, it seems to be potentially an effective method to predict the Fourier coefficients, by giving them as input to the neural network. For every time t we calculate the Discrete Fourier Series (DFS) of the points $x(t-T+1), ..., x(t)$, to obtain DFS coefficients $X_0(t), ..., X_{T-1}(t)$. These represent the result of a moving window of the DFS calculation. We train a separate neural network for every DFS coefficient. The network takes previous values of the DFS coeficient time series as inputs (that is $X_n(t-T_p+1), ..., X_n(t)$), and is trained to estimate the future coefficient $X_n(t+1)$. Once all T forecasts from the T neural networks are available, they are inverted to obtain an estimate for the whole signal in the T-long period from $t-T+2$ till $t+1$. The estimate of the signal at the end of the period (at time $t+1$) is taken as the signal forecast.

In addition to these methods, we considered the problem of fore-

casting several steps away (the multistep ahead problem). Thus, we have a time series $x(1), ..., x(t)$, and would like to forecast $x(t+k)$, where $k > 1$. Of course the larger k is, the more difficult the problem is. We have considered several methods to implement the multistep ahead forecast problem, and we will compare between these methods. The methods are:

Method 5) Direct Method: We train the neural network to directly forecast the k^{th} period ahead. Thus, the desired output for the network will be $x(t+k)$.

Method 6) Recursive Method: We consider a network that forecasts a single step ahead, and apply this network recursively to forecast k step ahead. Thus, at any intermediate step the network will use some of the forecasts it obtained at previous steps as inputs. There are two basic methods to train such a network. We implemented these two methods in the comparison:

a) To train the network to perform simply a single step ahead forecast.

b) To consider a backpropagation through time scheme [10]. That is, we consider the k steps ahead forecast as the result of a cascade of identical networks, and train this composite structure of networks.

IMPLEMENTATION RESULTS:

We have performed simulations for the comparisons for each of the single step ahead group and the multistep ahead group. For each comparison we performed five different runs using five different combinations of some of the inputs described last section. Table 1 shows the results for the test period for the single step ahead case. One can see that the most basic method, forecasting the actual flow (Method 1) results in the best forecast accuracy. The results were generally consistent accross all five runs, meaning that the variation in error among the five runs was small. For the multistep group, we have performed both a two step ahead comparison and a three step ahead comparison. One can see

that the direct method (Method 1) is superior, especially for the three step ahead case (see Table 2 and Table 3 respectively). The recursive method trained using a backpropagation through time approach was much better than that trained to perform a single step ahead forecast.

Although the comparative performance of the different approaches is usually problem dependent, this comparison should give some insight, and is therefore an addition to other comparison studies such as [11].

Method	Avg(Test Error)	StDev(Test Error)
Method 1	0.213	0.006
Method 2	0.240	0.014
Method 3	0.290	0.011
Method 4	0.287	0.008

Table 1: Comparison of single step ahead methods: the average and the standard deviation of the testing error

Method	Avg(Test Error)	StDev(Test Error)
Method 5	0.315	0.007
Method 6a	0.685	0.184
Method 6b	0.310	0.008

Table 2: Comparison of multistep ahead methods: the average and the standard deviation of the testing error for the two step ahead case

Method	Avg(Test Error)	StDev(Test Error)
Method 5	0.374	0.021
Method 6a	0.623	0.164
Method 6b	0.489	0.080

Table 3: Comparison of multistep ahead methods: the average and the standard deviation of the testing error for the three step ahead case

ACKNOWLEDGEMENTS:

The authors would like to acknowledge the help of Dr. Ashraf Ghanem and of M. Bayoumi in supplying the data.

REFERENCES:

[1] **Proc. Neural Networks in the Capital Markets Conf.**, London Business School, London, U.K., October 1995.

[2] J. Connors, D. Martin, and L. Atlas, "Recurrent neural networks and robust time series prediction", **IEEE Trans. Neural Networks**, Vol. 5, No. 2, pp. 240-254, March 1994.

[3] H. Awwad, J. Valdes, and P. Restrepo, "Streamflow forecasting forHan River basin, Korea", **J. Water Resources Planning and Management**, Vol. 120, No.5, pp. 651-673, 1994.

[4] M. El-Fandy, Z. Ashour, and S. Taiel, "Time series models adoptable for forecasting Nile floods and Ethiopian rainfalls", **Bulletin American Meteorological Society**, Vol. 75, No. 1, pp. 1-12, January 1994.

[5] D. Burn and E. McBean, "River flow forecasting model for the Sturgeon River", **Hydraulic Engineering**, Vol. 111, No. 2, pp. 316-333, February 1985.

[6] N. Karunanithi, W. Grenney, D. Whitley and K. Bovee, "Neural networks for river flow prediction", **Computing in Civil Engineering**, Vol. 8, No. 2, pp. 203-219, April 1994.

[7] R. Jacobs and M. Jordan, "A competitive modular connectionist architecture", in **Advances in Neural Information Processing Systems 3**, R. Lippmann, J. Moody, and D. Touretzky, Eds., pp. 767-773, Morgan Kaufmann, San Mateo, CA, 1991.

[8] S. Haykin, **Neural Networks: A Comprehensive Foundation**, IEEE Press, 1994.

[9] A. Weigend, M. Mangeas, and A. Srivastava, "Nonlinear gated experts for time series: discovering regimes and avoiding overfitting", **Int. Journal of Neural Systems**, Vol. 6, pp. 373-399, 1995.

[10] P. Werbos, "Back-propagation through time: what it does and how to do it", **Proceedings of the IEEE**, Vol. 78, No. 10, October 1990.

[11] A. Weigend and N. Gerschenfeld (Eds.), **Time Series Prediction: Forecasting the Future and Understanding the Past**, Santa Fe Institute, Addison-Wesley, 1994.

ROBUSTNESS OF A CHAOTIC MODAL NEURAL NETWORK APPLIED TO AUDIO-VISUAL SPEECH RECOGNITION

Harouna Kabré

CLIPS-IMAG Laboratory
Joseph Fourier University
Grenoble, BP 53, Bat B, 38041, France.
e-mail: Harouna.Kabre@imag.fr

Abstract

We stabilized a chaotic Modal Neural Network (MNN) for the purpose of robust speech recognition. A Modal Neural Network is an Artificial Neural Network system which includes two levels of information processing. The first level is trained to store and retrieve some acoustic and visual patterns. The different states of this network, which represent the sound classes in a task of speech recognition, are called modes and are supposed to chaotically evolve when speech recognition is performed in adverse environments.
The control of the chaotic behavior of the different modes constitutes the second level. An external signal, taken from a visual input such as the lip-opening parameters of the speaker is applied to stabilize an acoustic modal network of which the modes are moved from an initial position to a target position. The addressed task is the audio-visual recognition of the 10 French vowels, perturbed by some noises. The Perceptual Linear Predictive analysis applied to the speech signal of the 10 vowels outputs some vectors formed by 5 spectral parameters. They are in turn fed into a Modal Neural Network implemented as a feed-forward network. When the noise level increases, the classes stored by the acoustic MNN exhibit a chaotic behavior which is stabilized by the signal given by the visual path. We show that in an uncooperative environment, a chaotic modal neural network stabilizes well.

Key Words: Robustness, Chaos, Audio-Visual Speech Recognition, Adaptation.

0-7803-4256-9/97/$10.00

1. INTRODUCTION

Robustness is the ability for a given system to face unknown situations. In the speech domain the problem for a speech perceiver (e.g. an Automatic Speech Recognition System) is keeping good performances of speech recognition even if the spectral and temporal information conveyed in the speech signal are perturbed by some undesirable noises. This is a crucial problem because, when the training and testing conditions differ, the performances of the speech recognition systems can be drastically reduced, thus limiting the spread of speech technology to real world applications.

Even if the current approaches (which are based on the improvement of the quality of microphones or on a better speech signal processing) have given encouraging results [1], they are all based on the idea that the performance degradation of speech perceivers could be eliminated by decreasing the mismatch between their training and testing conditions. However, our experience in this domain [2] showed that only small changes in acoustic environments could be solved by this kind of approach, while big changes required more study on the understanding of the perturbation nature and learning about the impressive robustness of living systems.

Stein and Meredith (1993) demonstrated that the information processing in a cat's brain are initially segregated at the neural level. Neurons dedicated to one sense do not interact with those of another sense until the stimulus is transmitted to the brain. There the signal converges to the same target in the superior colliculus. The superior colliculus appears to be responsible for attentive and orientation behaviors. Based on these neurological facts two levels of information processing are considered in this paper.
The first level extracts modes from the global stimulus reaching the acoustic sensor (i.e. microphone). The different modes extracted constitute some stable and observable states of the acoustic sensorial subsystem and can be identified to one of the different classes of sounds in a given vocabulary of automatic speech recognition system. The second level processes the modes which could behave chaotically if the patterns encountered at the training and testing conditions are so different. The oscillations and chaos are ubiquitous in the brain. They reveal an indecisiveness between the input pattern and the stored patterns and it is guessed by some authors that oscillations and chaos may play an important role on binding and integration of different information in the brain. Moreover, some other findings in the speech transmission domain have shown that a chaotic system like the Lorenz system defined by a set of differential equations, is able to provide some self-synchronization ability to a speech transmitter and a speech receiver, so that their robustness increases [4].
Hence the oscillations and chaos of a Modal Neural Network could be used to increase the robustness of a speech perceiver.

Several methods have been developed for controlling the chaos [5] by which an unstable orbit can be stabilized in a chaotic system. Recently some authors have used them to control neural networks states by implementing an external artificially generated signal [6].
In this paper, the chaos of MNN is controlled by an external signal measured in a real world (i.e. by a visual input). The MNN in a chaotic state means that the stimulus provided by the acoustic path is not enough to suppress the indecisiveness of the input sound class. By using the modes extracted from the visual path as control signals, a stabilization occurs.

In the next section, we describe the general features of a MNN. In section 3, the network controller is introduced. In section 4 our experiment protocol and results are discussed. In section 5, we compare the results to others and conclude the paper in section 6.

2. MODAL NEURAL NETWORK

In this study, we assume two kinds of perturbations which in turn correspond to two kinds of robustness problems for a given information integration system:

- coherent perturbations can be solved by a control theory approach. In this framework some corrections of the system states could be obtained by designing a closed loop to drive the modes of the system. For us, this case corresponds to a testing condition not so different than the training conditions.
- chaotic perturbations require that the system be able to visit some new stable states which are very far from the conditions encountered during the training phase. In other words the system must be "creative" because it reflects the states of a creative system ("human talker"). Thus, the study carried here could be seen as a complementary vision of the robustness problem as discussed in [2].

We associate those two different perturbations to *kind* and *aggressive* robustness problems respectively. Kind robustness can be achieved by using the Kalman filtering or any other error correction method. Aggressive robustness requires a more "creative" behavior of the control system like the chaotic one. For the task considered in this paper, we model kind robustness with the Artificial Neural Networks which are considered to be noise-resistant, and specifically with Modal Neural Networks while aggressive robustness is modeled by the chaotic control of the MNN modes.

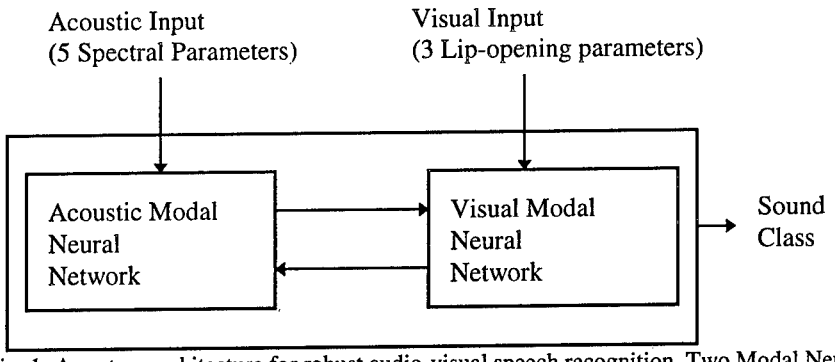


Fig. 1: A system architecture for robust audio-visual speech recognition. Two Modal Neural Networks (see text) model the information processing subsystems for the visual and acoustic paths.

A MNN can be taken as any neural network implementation of ANN to store, retrieve and manipulate some stimuli. In the speech domain a MNN for audio-visual speech perception takes as input a speech signal together with some lip-opening parameters and thus outputs the class of the sound: it implements a sound classifier.

Let I represent the input space and O the output space of data recorded from a system for which we try to find a model. The solution obtained with an Artificial Neural Network can be understood as the search of a function $f(W,I,E)$ which associates I to O in terms of information encoding into the weights W of the neural network. If the network is successfully implemented and trained then we can consider that is has computed an inverse matrix A so that we have: $O = A^{-1}I$. The modes of this latter matrix store the underlying mapping between I and O. Different solutions exist to solve this problem, e.g. the Hopfield network which seems to be plausible in terms of biological implementation. They are well-known examples. Some other solutions like feed-forward neural networks could be used to obtain the function f and the different modes.

3. CONTROLLING THE MODES OF A MNN

From the discussion in section 2, we will model a MNN as $S = \{\lambda_1, \lambda_2, \cdots, \lambda_i, \cdots, \lambda_M\}$, a system S having M modes $\lambda_i \; i = 1 \cdots M$. Let us denotes $m(t) = \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_M \end{bmatrix}$.

The problem is now to control the movements of the different modes so that they go towards some desired targets. When the modes are close to the unit circle in the Nyquist plane, then there are some oscillations. When the modes are real and positive there is instability and chaos.

We will use a parallel updating rule introduced in [3] and defined as:

$$m(t) = 1 - 2\Psi\left(a_1 m(t) + a_2 [m(t)]^2 + I(t)\right) \text{ with } \quad \Psi(z) = \frac{1}{\sqrt{2\pi}} \int_{z/\sigma}^{+\infty} e^{-x^2/2} dx$$

where $I(t)$ represents a control signal and a_1, a_2 two weighting parameters.

This rule generates phenomena like periodic, chaotic and bifurcation behaviors for different values of the noise level σ [6]. The control consists in taking the modes computed in the visual path as $I(t)$ and to recover clean acoustic modes from noisy ones.

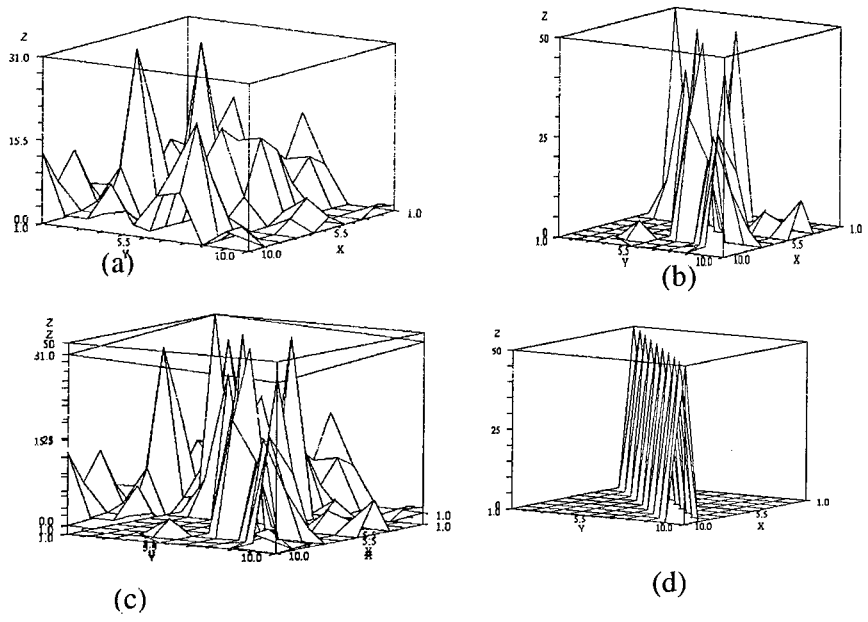


Fig. 2 : The modes computed on the audio path perturbed by a gaussian noise with a SNR of -20dB are chaotic (a). Conversely the visual path has less perturbed modes (b). The stabilization due to the visual path modes occurred in (d) after a period of indecisiveness in (c).

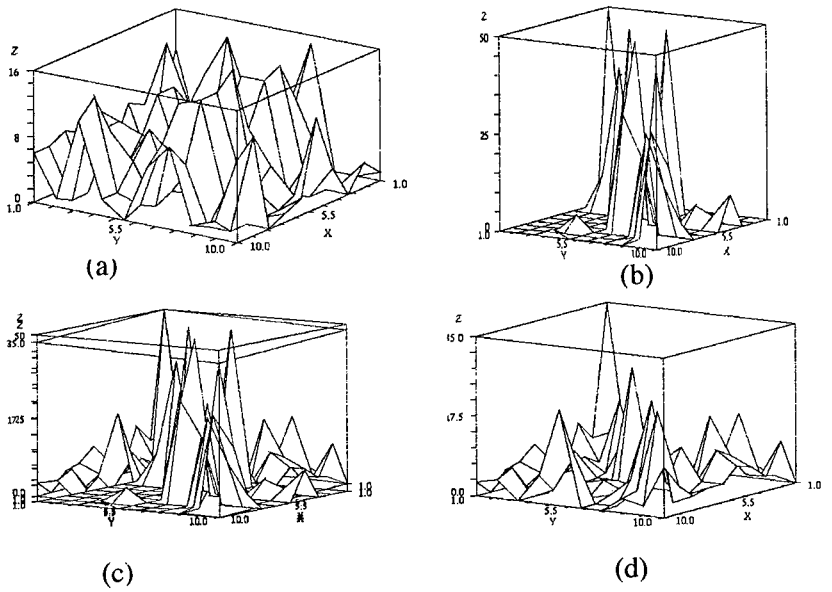


Fig. 3 : The modes computed on the audio path perturbed by another vowel sound with a SNR of -20dB are chaotic (a). Conversely the visual path has less perturbed modes (b). The stabilization due to the visual path modes occurred in (d) after a period of indecisiveness in (c).

4. EXPERIMENTS

The movie of a speaker pronouncing the 10 French vowels is considered the experimentation start point.
The image of the speaker's face is processed by the Snakes method to extract some lip-contours [7]. We applied this method to extract some lip-opening parameters (the center of gravity of the mouth opening area, the vertical and horizontal lip width). Those three lip parameters are input to the visual MNN.
Similarly, each vowel acoustic signal is processed with the Hermansky PLP method [8]. The resulting 5-dimension vector is given to the acoustic MNN.
For each vowel 100 pairs of audio-visual patterns are obtained for the experiment.

The experiment concerns the control of the modes computed on the auditory path by the visual path. The MNNs (acoustic and visual) are two feed-forward neural networks [9] which are trained to classify the 10 vowels. After the training the MNNs weights are frozen. For each MNN we do not make the sigmoidal thresholding at the

output units. Thus they have outputs varying between 0.0 and 1.0 corresponding to the strength of the network modes. For more discussions on the training of these networks see [9].
The first experiment concerns the audio-visual recognition of the 10 vowels perturbed by a gaussian noise (see Fig. 2) and the second for a perturbation by the vowel /a/ signal pronounced by a female speaker. Note that this latter case corresponds to a situation for which two speakers speak at the same time (i.e. the perturbation is the voice of another speaker). The two experiments are carried out for a Signal to Noise Ratio (SNR) of -20 dB.

In a quiet environment the acoustic input allows a good computation of the modes of the 10 vowels which would be on the diagonal like in Fig. 2d. When the noise level increases (see Fig. 2a), the coordinates of the modes behave chaotically because no sufficient information is delivered by the auditory path. This unstable behavior can then be controlled by the visual path which brings the complementary information (Fig. 2c) to stabilize the acoustic modes (see Fig. 2d).
If we change the kind of perturbation we observe similar results. However the effect of the visual path in the modal space is not as good (Fig. 3c), and this is due to the great perturbation of the vowel spectral information when the perturbation is a vowel sound. In the two cases we achieved the stabilization.

5. DISCUSSION

Many studies have been carried out on the control of the chaos. Among them some have specifically been applied to speech domains. Cuomo and al. (1993) have shown the interest of using Lorenz chaotic system for a robust transmission between a transmitter and a receiver. The basic idea is to take advantage of the self-synchronization feature of a chaotic system which is able from any starting initial condition to reach a given target. When a receiver and a transmitter lose their synchronization because of some perturbations in the channels, then the two chaotic systems are able to re-synchronize by themselves. The application consists in considering the speech signal to be transmitted as a low-level perturbation and this increases the reliability and robustness of the transmission. In this study this feature is used to move a MNN toward a nominal performance state.
The control of a Hopfield neural network has been shown in [6]. It showed a possibility to use a chaotic system as a two-module information processing system. The second neural system, identified as the colliculus in the cat's brains, sends a signal which stabilizes the chaotic behavior of the first neural system which stores and recalls patterns. Only some simulated data have been tested.
As far as we know, our present study seems to represent the first attempt to position the modes computed from audio-visual data recorded by a human subject.

6. CONCLUSION

We described a framework for controlling the chaotic movements of the modes of a neural network which learns to recognize visual speech. An external signal taken from another sensorial input (visual) allowed us to stabilize the network. The results obtained on the 10 French vowels displayed in a modal space and perturbed by a white noise (gaussian) and a colored noise (another vowel sound) has shown some strong capabilities of the Modal Neural Network for processing new incoming events occurring in real world conditions. The complementarity between the auditory and the visual paths is taken at the level of modes whereas the environment has been supposed to be handled by the MNN. The general methodology based on two interacting information processing modules could be applied to other kind of sensor integration problems and related to other studies on multi-modal data analysis.

If the basic structure of the MNN is a Hopfield network then our study may be compared to some biological plausible implementation of information processing in the brain. Nevertheless, the methodology developed could be taken as a contribution to the modeling of robust information processing for real world applications.

7 . REFERENCES

[1] Gong, Y., "Speech recognition in noisy environments: A survey", J. of Speech Communication., 16, pp. 261-291, 1995.

[2] Kabré, H, "On the Active Perception of Speech by robots", IEEE/RJ, Multi-Sensor Fusion and Integration for Intelligent Systems, Washington, pp. 765-774, 1996.

[3] Stein, B. E. & Meredith, M. A., "The Merging of the Senses", Cambrige, M.I.T press, 1993.

[4] Cuomo K., M., Oppenheim, A. V. & Strogatz, S., H., "Robustness and Signal Recovery in a Chaotic System", Int. J. Bifurcation and Chaos, Vol. 3, 6, pp. 1629-1638, 1993.

[5] Shinbrot T., Grebogi C., Ott, E., Yorke J. A., "Using small perturbations to control chaos", Nature, Vol. 363, 411-417., 1993.

[6] Li, Y. Y., Zheng N. N. & Yuan L. X., "Controlling the chaotic neural network, a way of information integration", IEEE/RJ, Multi-Sensor Fusion and Integration for Intelligent Systems, Washington, pp. 775-780, 1996.

[7] K. Kroschel, K., Mekheil, M., S., Kabré, K., "Modeling the Lip-Contour in Noisy Images for Lip-reading Applications", Institut Franco-Allemand d'Automation, Karlsruhe, 1996.

[8] Hermansky, H., "Perceptual Linear Predictive (PLP) Analysis of Speech", J. Acoust. Soc. Am, 87,4, PP. 1738-1752, 1990.
[9] Kabré, H., "A Probabilistic Model for Multi-sensor Data Fusion"., J. of Speech Communication, (to be published), 1995.
[10] Junqua, J. C., "Acoustic and production pilot Studies of speech vowels produced in noise", Proc. Of Spoken Language, pp. 811-814, 1992.
[11] Mansour, D. & Juang, B. H., "A family of distortion measures based upon projection operation for robust speech recognition", IEEE Internat.. Acoust. Speech Signal Process, NewYork, pp. 36-39, 1988.
[12] Bateman, D. C., Bye, D. K. & Hunt M. J., "Spectral normalization and other spectral technics for speech recognition in noise", Proc. Of The IEEE Inter. Conf. On Acoustic Speech Signal Processing San Francisco, Vol. I, pp. 241-244, 1992.
[13] Petajan, E. P., "Automatic Lip Reading to Enhance Speech Recognition", In Proceedings IEEE Common Society Global Telecom Conf., Atlanta GA, 1984, 265-272, 1984.
[14] Massaro, D. W., "Speech Perception by Ear and Eye: A Paradigm for Psychological Inquiry", Lawrence Erlbaum, Associates London, 1987.
[15] Yuhas, B. P., Goldstein, M. H. & Sejnowski, T. J., "Interpretation of Acoustic and Visual Speech Signal using Neural Networks". IEEE Common Magazine, 1989.
[16] McGurk, H., MacDonald, J., "Hearing Voices and Seeing Eyes", *Nature, 264, 746-748*, 1976.
[17] Summerfield, Q., Audio-visual Speech Perception by Lip-reading and Artificial Simulation", in Hearing Science and Hearing Disorders, M.E. Putman and M. P. Haggard eds, Newyork, Academic press, 1983.
[18] Summerfield Q. , Some Preliminaries to a Comprehensive Account of Audio-Visual Speech Perception. in B. Dodd and R. Campbell eds, Hearing by eyes: the psychology of lip-reading, pp 3-51, London: Lawrence Erlbaum associates, 1987.
[19] Brooks, R., "Intelligence without representation", Artificial Intelligence, vol. 47, pp. 139-159, 1991.
[20] Rumelhart, D. E., Hinton, G. E., R. J. Williams, R. J., "Learning internal representations by error propagation", in Parallel Distributed Processing. Cambridge MA : MIT Press, 8, pp. 318-362.
[21] Varela, F., "Principles of Biological Autonomy", New York, Elsevier North Holland, 1979.
[22] Kabré, H. & Vigouroux, N., "A Lateral Inhibition Neural Network for the Continuous Speech Decoding", Advances on Modeling and Simulations for the Enterprises, A., AMSE Press, Vol 11, no 2, pp. 13-23.

APPLYING NEURAL NETWORKS AND OTHER AI TECHNIQUES TO FAULT DETECTION IN SATELLITE COMMUNICATION SYSTEMS

Liese Elerin, Charles Learoyd, and Beth Wilson
Raytheon Company
1001 Boston Post Rd.
Marlboro, MA 01752
bwilson@ed.ray.com

ABSTRACT

A demonstration program has recently been completed to apply various artificial intelligence techniques including neural networks, expert systems, and case-based reasoning to fault detection in satellite communications systems. The GMM program implemented these techniques for Global Military Satellite Communications Maintenance. Neural networks were designed and trained to analyze incoming Built-In-Test (BIT) fault signatures from the satellite communications terminal. Expert systems were developed to embed diagnostic knowledge relating to equipment maintenance. The prototype hybrid system uses neural filters to detect faults, which are further processed by expert systems to classify the faults and provide repair directions.

INTRODUCTION

Present military communications systems perform Built-In-Test (BIT) false alarm reduction and fault isolation through largely heuristic algorithms such as n-of-m filtering and deterministic isolation trees. These methods are often implemented in a combination of system processing (typically software) and manual procedures contained in maintenance manuals. Furthermore, methods of collecting BIT history often rely on the maintainer to fill out paperwork, with no systematic feedback.

Once fielded, BIT performance of these systems has proven difficult to improve, both because upgrades are difficult and because of the lack of centralized analysis and machine learning. Consequently excessive unnecessary repair actions and the need for skilled maintainers keep maintenance costs high over the life of the system.

GMM AI-BASED APPROACH TO BIT

The purpose of the Global MILSATCOM Maintenance (GMM) program was to apply Artificial Intelligence and Neural Network methods to address the lack of automated systematic improvement BIT performance, especially in the post-fielding environment.

In the GMM concept, terminals process BIT fault signatures using AI techniques. Whenever a maintenance action occurs, history is automatically transmitted to a central analysis facility using satellite communication resources on an as-available basis. At the central facility, a diagnostic/system analyst uses AI tools to analyze the accumulated history and, if appropriate, generate updates to the AI processing in the fielded terminals. Hence, each fielded system benefits from the cumulative experience of all fielded systems.

GMM has the potential to significantly reduce fielded system maintenance costs. The savings can arise from improved fault diagnosis which allows operators to assume field maintenance duties, thereby eliminating skilled field maintainers and related training and personnel costs, and reducing unnecessary repair actions.

The goal of the GMM program was to demonstrate the GMM concept by developing and integrating prototype components employing the AI techniques of neural networks, expert systems and case-based reasoning.

Figure 1 graphically depicts the system concept. In this concept, the fielded terminals may have neural network filters at several levels in the equipment hierarchy: module, assembly, subsystem and system. These neural filters may receive corroborating fault signature data from BIT hardware and other neural filters and also environmental information such as temperature or vibration. They use this data to make local low-level fault classifications. The results of the neural filters may be used by an expert system to make higher-level fault classifications and give repair directions.

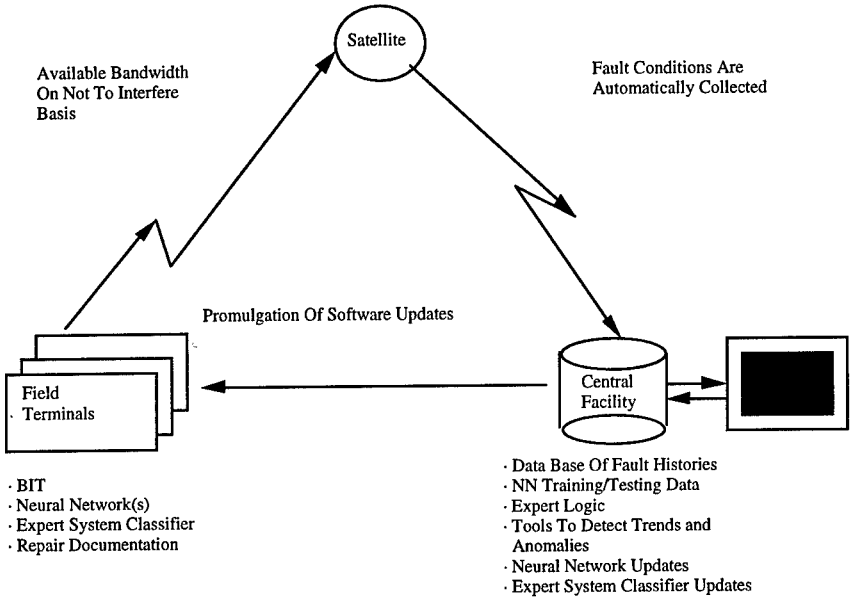


Figure 1. GMM System Concept

Reports on BIT fault indications are sent to a central facility and stored. Fault signatures, environmental information, repair actions and fault isolation decisions are also sent, using the communication system on a low priority "as-available" basis. In the conceptual GMM system, the central site has a storage capability for fault history data from the terminals. An analyst reviews the data to determine if the neural networks or the expert system logic are incorrect. The fault isolation logic can be modified and new test cases can be added to the training data library of the neural networks. Periodically the fault isolation logic and the neural networks can be updated and distributed.

The use of a centralized AI-based analysis facility with automated, user-transparent access to fault and repair data from a large community of fielded systems which themselves employ AI techniques is the key to effectively introducing AI and learning techniques at the organizational maintenance level.

In the GMM concept, neural network technology was selected for use at terminal field sites to refine and improve the results of BIT fault reporting and subsequent fault detection. The results of this development are described in this paper.
Expert system technology was selected to improve the knowledge acquisition process as well as to provide the capability to generate executable fault isolation logic. In concept, the diagnostic system will consist of two tools, one for diagnostic fault logic development and one for execution of that logic.

The technology of case-based reasoning (CBR) was selected for use in maintaining and optimizing parts of the system which can be more costly to maintain traditionally, such as the expert system. In addition, CBR embodies an inherent mechanism for system adaptation and learning. The case-based reasoning function examines the failure events, compares them to known cases in the library, and recognizes deficiencies. Also, since the incoming failure events may not exactly match the existing cases in the library, the case-based reasoning function will incorporate new experiences into the library by adapting existing cases with information from the new event to make new cases. The case library will grow with new information, which can then be used by the system analyst to determine where and when to update the other functions of the system. The global position of the case library and case-based reasoning function will allow the distributed knowledge of the system to be collectively examined, refined, and returned to all sites consistently. After the concept was defined, funding cuts did not permit this CBR component to be implemented in the prototype.

NEURAL NETWORK DEVELOPMENT

We selected a representative terminal system with BIT based on traditional technology. The system was comprised of a rich set of replaceable units. These replaceable units are noted in this paper as generic components labeled units A-E as depicted in figure 2.

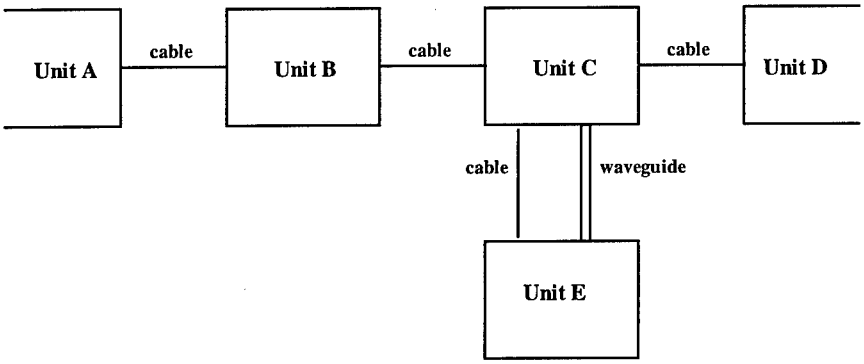


Figure 2: Representative Terminal System

The fault signatures available can be represented in the following ambiguity groups:

- a subcomponent of Unit B
- a subcomponent of Unit C
- the cable between Units A and B
- the cable between Units B and C
- the cable between Units C and D
- the cable between Units C and E
- the waveguide between Units C and E
- a subcomponent of either Unit B or Unit C, but not the cable between Units B and C

The BIT signatures for the Unit B/Unit C have 96 bits in their raw form. Examination of the Unit B/Unit C fault logic trees revealed that the terminal isolation processing uses less than half of the fault bits. Those bits that did not contribute to a decision (i.e., designator bits, RF switch status, etc.) were removed from consideration, leaving 39 relevant bits.

Next, a matrix which we called the Base Vector Table (BVT) was constructed with a column for each of the 39 bits and a row corresponding to each indictment as defined by each distinct solution node of the fault logic trees. The BVT provided a tabular representation of the terminal's fault isolation processing for the Unit B and Unit C.

After the input data sources were identified and examined, a set of network output classifications was defined:

1. Unit A/cable/Unit B
2. Unit B
3. Unit B/cable/Unit C
4. Unit B/Unit C
5. Unit D/cable/Unit C
6. Unit C
7. waveguide/Unit E/Unit C
8. Unit E/cable/Unit C
9. No Fault

From the set of 9 output classes, two others were derived. The first consisted of 4 classes and grouped all Unit B indictments, all Unit C indictments and all Unit B/Unit C indictments together:

1. Unit B (from faults 1 and 2 above)
2. Unit C (from faults 5, 6, 7, and 8 above)
3. Unit B/Unit C (from faults 3 and 4 above)
4. No Fault (from 9 above)

The second consisted of a binary "yes/no" classification, based on the given unit. This required two binary classifiers, one for the Unit B and one for the Unit C. The binary Unit B classifier had the following classes:

1. Unit B (from faults 1, 2, 3, and 4 above)
2. Not Unit B (from faults 5, 6, 7, and 8 above)

The binary Unit C classified had the following classes:

1. Unit C (from faults 3 through 8 above)
2. Not Unit C (from faults 1, 2, and 9 above)

The motivation for the binary network architecture was to configure the networks as binary decisions with an overall classification scheme resembling a binary decision tree. By structuring the individual neural networks as binary decisions with one output node, the network can effectively be "tuned" to detect the desired classification. As a detector, well-established statistical methods can be applied to better understand the data and the network. These methods are discussed in detail in Appendix A which describes the Multi-layer Layer Perceptron (MLP) design algorithm [1].

This algorithm uses information from the training data set to structure the network, including calculation of the necessary number of hidden nodes to execute the mapping, and computation of "good" starting weights. The algorithm provides a means of designing portions of the architecture of the neural networks in advance. It can also reduce training time, improve the convergence rate, increase the likelihood of convergence at a global minimum, and improve the generalization performance on novel data.

NEURAL NETWORK RESULTS

The failure data collected during factory testing demonstrated the potential of pre-deployment "field" data to provide early feedback to a central facility in a GMM system, or to system engineers in a more traditional design environment. However, to be effective, all opportunities to collect BIT signature data should be exploited. In addition, a rigorous and preferably automated method of linking repair actions to failures and BIT signatures must be in place.

A summary of the training/testing results for all networks is given in Table 1. The fact that the training and testing results were similar indicated that the networks were able to generalize and were not "memorizing" the training data. The results indicated that the networks could distinguish well among the different classes.

Network	% Correct Synthesized Training Data	% Correct Synthesized Testing Data	% Correct Factory Data
9-Class Heuristic	100.0	98.4	100.0
4-Class Heuristic	99.4	99.0	96.4
Binary Unit B Statistical	100.0	95.9	96.4
Binary Unit C Statistical	99.5	91.5	93.0

Table 1. Results of Network Training and Testing

ADVANTAGES OF THE GMM APPROACH

The major advantage of the GMM approach to maintenance is the automated, closed-loop nature of the data collection, analysis, and feedback path. This allows each fielded system to benefit from the cumulative experience of all fielded systems.

Fault information is processed using a variety of AI techniques, each selected for its particular suitability to the task it performs. Whenever a maintenance action occurs at a given field site, history is automatically transmitted to a central analysis facility where a diagnostician or system analyst uses AI tools to analyze the accumulated history and, if appropriate, generate updates to the AI processing in all the fielded systems. In addition, the automated nature of the process provides a much-needed consistency in the methods of making and installing modifications to the diagnostic tools and process.

The capability of evolutionary improvement was experienced first-hand by the GMM neural network developers. During network training it was discovered that the "No Fault" data class had not been included in the training data for the statistical networks. As a result, their classification results were initially poor. Examples of "No Fault" data were added to the training data set and the networks were retrained and tested. As expected, a second iteration of testing against the factory data yielded much improved results as shown in Table 2. This is an example of the GMM concept in practice.

Network	**% Correct Factory Data Without "No Fault"**	**% Correct Factory Data With "No Fault"**
Binary Unit B Statistical	75.0	96.4
Binary Unit C Statistical	64.0	93.0

Table 2: Improved Results by Adding "No Fault" Class

SUMMARY

The major conclusions of the project were that the study of the neural networks indicated that the backpropagation network model has potential for application to fault location. The expert system tools KAS and LTS were found to be well-suited for inclusion in a GMM system. Additional work is needed to prototype an implementation of the Central Analysis Facility in order to confirm the effectiveness of the overall GMM concept. Enhancements to the GMM concept were identified that would broaden its scope beyond fielded systems to include depot facilities, an automated hardware component tracking system, and a maintainer training system.

The work described in this paper was performed by Raytheon Company under the Global MILSATCOM Maintenance (GMM) contract for the Defense Advanced Research Projects Agency (DARPA) with U. S. Army CECOM, PM SATCOM as Executive Agent [3]. It expanded upon work performed under a previous contract, Neural Network False Alarm Filtering, which was performed by Raytheon Company for Rome Laboratory Air Force Materiel Command [4].

REFERENCES

[1] E. Wilson and D. Tufts, "Multilayer Perceptron Design Algorithm," Proceedings of the 1994 IEEE Workshop on Neural Networks for Signal Processing, pp. 61- 68.

[2] F. Aylstock, L. Elerin, J. Hintz, C. Learoyd, and R. Press, "Application of Neural Network Technology to Built-in Test False Alarm Filtering," Ninth AIAA Conference on Computers in Aerospace, San Diego, CA October 1993.

[3] Scientific and Technical Report for the Global MILSATCOM Maintenance (GMM) Program, Volume I, September 1996 (pending authorization for public release).

[4] RL-TR-94-216. Neural Network False Alarm Filter Final Technical Report. December 1994, Volume I.

Multi-linguistic Handwritten Character Recognition by Bayesian Decision-based Neural Networks*

Hsin-Chia Fu† Y.Y. Xu
Department of Computer Science and Information Engineering
National Chiao Tung University
Hsin-Chu, Taiwan, ROC
Email: hcfu@csie.nctu.edu.tw
Fax: (886)-3-5724176, Tel: (886)-3-5731930

Abstract

This paper proposes a multi-linguistic handwritten characters recognition system based on Bayesian decision-based neural networks (BDNN). The proposed system consists of two modules: First, a **coarse classifier** determines an input character to one of the pre-defined subclasses partitioned from a large character set, such as Chinese mixed with alphanumerics. Then a **character recognizer** determines the input image to its most matched reference character in the subclass. The proposed BDNN can be effectively applied to implement all these modules. It adopts a hierarchical network structures with nonlinear basis functions and a competitive credit-assignment scheme. Our prototype system demonstrates a successful utilization of BDNN to handwriting of Chinese and alphanumeric character recognition on both the public databases (HCCR/CCL for Chinese and CEDAR for the alphanumerics) and in house database (NCTU/NNL). Regarding the *performance*, experiments on three different databases all demonstrated high recognition (88~92%) accuracies as well as low rejection/acceptance (6.7%) rates, as elaborated in Section 3.2. As to the processing speed, the whole recognition process (including image preprocessing, feature extraction, and recognition) consumes approximately 0.27second/character on a Pentium-90 based personal computer, without using hardware accelerator or co-processor.

1 Introduction

Machine recognition of characters has been a topic of intense research since 1960's [4]. During the last decade, more and more commercial products were

*This research was supported in part by the National Science Council under Grant NSC 85-2213-E009-125.

†The corresponding author.

0-7803-4256-9/97/$10.00 ©1997 IEEE

available in the market. On the other hand, due to the enormous number of variations involved, handwriting recognition applications still require more work before they can reach comparable performance as a human. Generally speaking, current handwriting recognition techniques and problems can be categorized into the following two types: (1) on the character input methods (e.g., on-line and off-line approaches), and (2) on the character languages.

On-line and off-line: For the on-line approach, a computer receives trajectory coordinates by sampling the writing trace from a pen-based panel or a tablet. For the off-line approach, the whole text image is scanned into a computer and is stored in digital image format. Therefore, off-line character recognition process are usually more difficult than on-line approaches. Nevertheless, both on-line and off-line recognition techniques have their unique technical problems.

Language issues: It is well known that Chinese characters (including Kanji in Japanese) are unique and different from that of western languages, in that they are non-alphabetic and have quite complicated stroke structures. Taking modern Chinese language as example, there are more than 5,000 commonly used characters and a word may consists of one or several characters. The large character set of a language usually causes degradation in recognition accuracy and speed.

In addition, directly applying current mono-language character recognition techniques [2] to multi-linguistic document will be quiet difficult, due to:

1. separating mixed characters of different languages efficiently and correctly is not trivial, sometimes it is just as hard as recognizing characters,

2. implementing two or more different types of recognition modules is not time and space (both software and hardware) efficient,

3. combining recognition results from two different types of recognition modules is somewhat an unnecessarily extra work.

Therefore, it is desirable to design a uniform recognition architecture for the multi-linguistic character recognition. First, select a set of proper character features for characters of different languages in a document, such that they can be represented by a uniformed feature vectors. Then, a character recognition architecture for large character set can be adopted directly (or with just minor modification) to multi-linguistic character recognition. Comparing to the large character set such as Chinese, alphanumerics can be considered as a small set of special characters to the larger Chinese character set. Thus, uniformed feature selection and recognition architecture can be applied.

It is well known in the statistical pattern recognition [1], that the Bayesian decision rule can be implemented as an optimal image pattern classifier. By applying the statistical features of a character image, the Bayesian rule can

be used to match an input character to a reference character with minimal classification error. A Bayesian Decision-based Neural Network (BDNN) has the merits of both neural networks and statistical approaches. For example, Lin and Kung [3] proposed a probabilistic Decision Based Neural Networks (PDBNN) for the implementation of a face recognition/detection system. More specifically, the modularity of the BDNN makes itself suitable implementation for not only the mono-language, but also for the multi-linguistic language character recognition. Therefore, we propose the BDNN to attack the multi-linguistic character recognition problems. The organization of this paper is as follows: In section 2, the mathematical background and the architecture of the BDNN is presented. Then, the overview of a total system for handwriting recognition is presented in Section 3. The system consists of two modules, which are all implemented by the BDNN. The two modules: *coarse classifier* and *character recognizer* are discussed in great details. Experimental results of these modules are provided in both of these two sections.

2 Bayesian Decision-based Neural Network

Bayesian Decision-based Neural Network (BDNN) is a Bayesian rule based modular neural network for classification. One subnet of a BDNN is designed to represent one object class. Suppose there are k categories $\omega_1, \cdots, \omega_k$ in the feature space of a classifier or a cluster, the Bayesian decision rule classifies input patterns based on their posterior probabilities: An input character $\mathbf{x}$ is classified to category ω_i if $P(\omega_i \mid \mathbf{x}) > P(\omega_j \mid \mathbf{x})$, for all $j \neq i$. Suppose the likelihood density of input pattern $\mathbf{x}$ given category ω_i is a D-dimensional Gaussian distribution, the posterior probability $P(\omega_i \mid \mathbf{x})$ by Bayes rule is $P(\omega_i \mid \mathbf{x}) = \frac{P(\omega_i)p(\mathbf{x}|\omega_i)}{p(\mathbf{x})}$, $p(\mathbf{x} \mid \omega_i) = N(\mu_i, \mathbf{\Sigma}_i)$ where $P(\omega_i)$ is the prior probability of category ω_i ($\sum_{i=1}^{K} P(\omega_i) = 1$), and $p(\mathbf{x}) = \sum_{i=1}^{K} P(\omega_k)p(\mathbf{x} \mid \omega_k)$.

The category likelihood function $p(\mathbf{x} \mid \omega_i)$ can be extended to the mixture of Gaussian distributions. Define $p(\mathbf{x} \mid \omega_i, \Theta_{r_i})$ to be one of the Gaussian distributions which consist of $p(\mathbf{x} \mid \omega_i)$, $p(\mathbf{x} \mid \omega_i) = \sum_{r_i=1}^{R_i} P(\Theta_{r_i} \mid \omega_i)p(\mathbf{x} \mid \omega_i, \Theta_{r_i})$, where $P(\Theta_{r_i} \mid \omega_i)$ is the prior probability of cluster r_i, and $\Theta_{r_i} = \{\mu_{r_i}, \mathbf{\Sigma}_{r_i}\}$ is the parameter set for the cluster r_i, when input character patterns are from category ω_i. By definition, $\sum_{r_i=1}^{R_i} P(\Theta_{r_i} \mid \omega_i)$=1. In most general formulation, the basis function of a cluster should be able to approximate the Gaussian distribution with full rank covariance matrix, that

$$p(\mathbf{x} \mid \omega_i, \Theta_{r_i}) = \frac{1}{(2\pi)^{\frac{D}{2}} |\mathbf{\Sigma}_{r_i}|^{\frac{1}{2}}} \exp\left[-\frac{1}{2}(\mathbf{x} - \mu_{r_i})^T \mathbf{\Sigma}_{r_i}^{-1}(\mathbf{x} - \mu_{r_i})\right], \qquad (1)$$

where $\mathbf{\Sigma}_{r_i}$ is the covariance matrix. However, for those applications which deal with high dimension data but finite number of training patterns, the training performance and storage space discourage such matrix modeling. A natural simplifying assumption is to assume uncorrelated features of unequal

importance. Suppose that $p(\mathbf{x} \mid \omega_i, \Theta_{r_i})$ is a D-dimensional Gaussian distribution with uncorrelated features, that $\mu_{r_i} = [\mu_{r_i 1}, \mu_{r_i 2}, \cdots, \mu_{r_i D}]^T$ is the mean vector, and diagonal matrix $\mathbf{\Sigma}_{r_i} = diag[\sigma^2_{r_i 1}, \sigma^2_{r_i 2}, \cdots, \sigma^2_{r_i D}]$ is the covariance matrix. As shown in Figure 1, a BDNN contains K subnets, which are used to represent a K-category classification problems. Inside each subnet, an elliptic basis function (EBF) is used to serve as the discriminate function for each cluster r_i: $\phi(\mathbf{x}, \omega_i, \Theta_{r_i}) = -\frac{1}{2}\sum_{d=1}^{D} \alpha_{r_i d}(x_d - \mu_{r_i d})^2 + \theta_{r_i}$. If θ_{r_i} is set to $\theta_{r_i} = -\frac{D}{2}\ln 2\pi + \frac{1}{2}\sum_{d=1}^{D} \ln \alpha_{r_i d}$, then $\exp\{\phi(\mathbf{x}, \omega_i, \Theta_{r_i})\}$ can be viewed as the Gaussian distribution, as described in Eqn. (1), except for a minor notational change: $\frac{1}{\alpha_{r_i d}} = \sigma^2_{r_i d}$.

2.1 Learning Rules for BDNN

The training scheme for *BDNN* contains the following three phaseOBs: **unsupervised** learning phase, **supervised** training phase, and **self-grow** learning phase.

Unsupervised Learning Phase: The first phase of BDNN training is an unsupervised learning. The values of the parameters in the network are initialized in this learning phase. Many unsupervised clustering algorithms, such as K-means or Vector Quantization methods can be applied to unsupervised learning.

Supervised Learning Phase: As for the **supervised learning**, teacher information is used to fine-tune decision boundaries. When a training pattern is misclassified, the *reinforce* or *anti-reinforced* learning technique is applied:

$$\begin{aligned} \text{Reinforced Learning} &: \mathbf{w}^{(m+1)} = \mathbf{w}^{(m)} + \eta \nabla \phi(\mathbf{x}, \mathbf{w}) \\ \text{Antireinforced Learning} &: \mathbf{w}^{(m+1)} = \mathbf{w}^{(m)} - \eta \nabla \phi(\mathbf{x}, \mathbf{w}) \end{aligned} \tag{2}$$

Threshold Updating The threshold value of BDNN recognizer can also be learned by reinforced or anti-reinforced learning rules.

Self-growing Phase: One of the difficult in the unsupervised learning for BDNN is the decision on the selection of the proper number of clusters for the *K-means* algorithm. Therefore, we propose the self-growing of clusters (in the following, it is called receptor) during the supervised learning phase.

There are three main aspects for the self-growing rules:
(I1) When a new receptor should be created?
(I2) Which receptor should be partitioned to create a new receptor? and
(I3) How to initialize the centre and the covariance of the created new receptor?

On Issue I1, while the training sets are presented again and again, however the train status (especially the recognition rate) states unchanged or improves very slowly. In other words the current BDNN can not properly

learn and represent the whole training data sets. Therefore, extra Bayesian type receptors are needed to improve the modeling power of the current BDNN.

On Issue I2, when an extra receptor is needed to improve the training performance, we propose to create a new receptor from the receptor which produces the most of misclassification during the recent training processes. On Issue I3, when a new receptor is created, its initial values of the centre and covariance matrix needs to be properly determined, otherwise the classification capability may not be improved efficiently. Suppose that, a character pattern $\mathbf{x}$ corresponding to cluster Θ_i is presented to a BDNN classifier, then let us look into two receptors: the receptor S_i corresponds to cluster Θ_i, and the receptor (say S_j) corresponds to the largest response among the clusters other than Θ_i. Let o_i and o_j be the output for receptor S_i and receptor S_j, respectively. Let us define the ratio of these two outpus with respect to the training pattern $\mathbf{x}$ as $\rho_{\mathbf{x}} = \frac{o_j}{o_i}$. According to the retrieving scheme of the proposed BDNN, if o_j is larger than or equal to o_i (i.e., $\rho_{\mathbf{x}} \geq 1$), then the retrieving result of input pattern $\mathbf{x}$ must be wrong. Apparently, the smaller the $\rho_{\mathbf{x}}$ could be, the better the classification performance would be. Therefore, it would be better to initialize the new receptor with proper parameters (i.e., μ_i, Σ_i), such that the $\rho_{\mathbf{x}}$ has smaller values (≤ 1) with respect to all the other receptors S_j. Since a proper initial value μ_0 and $\mathbf{\Sigma}_0$ will have smaller $\rho_{\mathbf{x}}$, thus, the best position for the centre should be located at $\mathbf{x}$, i.e., $\mu_0 = \mathbf{x}$, so that the new receptor will generate the maximal output, for the input pattern $\mathbf{x}$. To determine the $\mathbf{\Sigma}_0$, we let $\mathbf{\Sigma}_0 = \sigma \mathbf{I}$, σ is a positive constant (to be determined). Suppose that the σ of the new receptor S_i is not properly determined, as shown in Figure 3(a), then the created receptor will have its largest possible output $o_i(\mu_i)$ to be smaller than the output $o_j(\mu_i)$ of an existing receptor S_j. Consequently, the new receptor S_i will not be able to reduce the value of $\rho_{\mathbf{x}}$. We call that the receptor S_i is "*overwhelmed*" by receptor S_j. To prevent the overwhelming problem, Figure 3 (b) shows a properly initiated new receptor S_i. Therefore, the following two constraints are suggested.

$$o_j(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{D}{2}}|\Sigma_j|^{\frac{1}{2}}} \exp\left[-\frac{1}{2}(\mathbf{x}-\mu_j)^T \mathbf{\Sigma}_j^{-1}(\mathbf{x}-\mu_j)\right] < o_i(\mathbf{x}) = \frac{1}{(2\pi\sigma)^{\frac{D}{2}}} \quad (3)$$

$$o_i(\mu_j) = \frac{1}{(2\pi\sigma)^{\frac{D}{2}}} \exp\left[-\frac{1}{2}(\mathbf{x}-\mu_j)^T \mathbf{\Sigma}_i^{-1}(\mathbf{x}-\mu_j)\right] < o_j(\mu_j) = \frac{1}{(2\pi)^{\frac{D}{2}}|\Sigma_j|^{\frac{1}{2}}} \quad (4)$$

These two constraints imply that receptor S_i and receptor S_j will not overwhelm each other. To satisfy Eq.(4), σ should be less than $\frac{1}{2\pi o_j(\mathbf{x})^{\frac{2}{D}}}$. Thus, by using $\frac{1}{2\pi o_j(\mathbf{x})^{\frac{2}{D}}}$ as an initial value, σ can be iteratively decreased by a small value η ($0 \leq \eta \leq 1$) until Eq.(4) is satisfied. Then, the final value of σ can be used as the initial value of σ_o for the new receptor S_i.

3 BDNN Handwritten Character Recognition System

A BDNN-based Handwritten character Recognition system is being developed in the Neural Networks Laboratory of National Chiao-Tung University. The total system configuration is depicted in Figure 2. All these three main modules, pre-processing and feature extraction module, coarse classifier, and character recognizer post-recognition module are implemented on a Pentium-90 based personal computer. A 300 dpi scanner is used to acquire document images, or page images. The acquired binary images are then down sized to 150 dpi for the following processing.

3.1 Image Pre-processing and Feature Extraction

Image pre-processing of a multi-linguistic character recognition is by no means of any different from the mono-language character recognition. The binary images of a handwritten character are first passed through a series of image processing stages, such as boundary smoothing, noise removing, space normalization, and stroke thinning operations. By evaluating most of the well known feature [2], we selected features with high index value, such as *crossing count* (CCT), *belt shape pixel number* (BSPN), and *stroke orientation feature* (STKO) as candidate features for the proposed character recognition system.

3.2 Multi-stage character recognition

Since there are as many as 5000 commonly used characters, and 62 of alphanumerics and symbols in a traditional multi-linguistic Chinese document, thus it is desired to perform a coarse classification (or clustering) to reduce the domain size of the *character recognition.* By having a small working domain, not only the over all recognition speed and recognition rate can be greatly improved, also the training on the fine-grained character recognizer can be much easier and faster. As illustrated in Figure 4, the two stage mixture character recognizer contains a coarse classifier as the first stage, then followed by a fine-grained character recognizer.

3.2.1 Coarse classification

In order to achieve a balanced recognition performance in a multi-stage recognition system, the coarse classifier needs to maintain a very high accuracy, (e.g., $\geq$ 99.9%). Although this is a difficult task, we propose to use the *CCT* feature and *overlapped K-means* clustering algorithm [2] for the implementation of the coarse classifier. By applying the two public databases suggested in Section 3.3, to train the proposed coarse classifier, this goal was achieved. The training and testing results are listed in Table 1.

Table 1: The training and testing results of coarse classification

Number of cluster	Ave. No. of characters in a cluster	Inside test of clustering rate	Outside test of clustering rate
60	1156	100 %	99.9%

3.2.2 Character recognition

The design of character recognizer is based on a BDNN for a character basis. A character recognizer is dynamically formed by a set of character BDNNs, which are one to one corresponding to the characters in an activated cluster by the coarse classifier. The training of the character BDNNs was first conducted by the *unsupervised learning* schemes, then each BDNN was fine tuned by the *supervised learning* scheme. During the retrieving phase, each of the character BDNN produces a score according to the mixture Gaussian distribution function $p(\mathbf{x} \mid \omega_i, \Theta_{r_i})$. The character BDNN with the highest score is the winner and its corresponding reference character is considered as the output of the character recognizer. By allowing about 6.7% of rejection on the input characters, the the recognition accuracy can reach 92.12%. Training and testing results of the character recognizer are listed in Table 2.

Table 2: Experimental results of the Multi-linguistic character recognition by BDNN.

Test type	Top 1.	Top 2.	Top 3.
inside	91.07 %	95.15 %	96.18 %
outside	88.22 %	89.76 %	90.89 %
w/rej	92.12 %	93.01 %	93.68 %

3.3 Handwritten Character Databases

In this research, there are two main sources of databases for the training and testing of BDNN: the CCL/HCCR1 [5] database and the CEDAR database. The CCL/HCCR1 database contains more than 200 samples of 5401 frequently used Chinese characters. The samples were collected from 2600 people including junior high school and college students as well as employees of ERSO/ITRI. Each sample character was scanned at 300dpi to generate a 144 × 150 pixel image. According to the script regularity, each character sample in the 5401 database was manually arranged in sequential order. In other words, suppose a handwritten character is scripted like a printed character, then it is placed at the beginning of the sequence. Consequently, cursive handwritten character samples will be placed at the bottom part of the sequence. The CEDAR database contains various style of handwritten alphanumerics, which were lifted from envelop address blocks in USA.

3.4 Overall performance evaluation

The system built upon the proposed has been demonstrated to be applicable under reasonable variations of character orientation, sizes, and stroke width. This system also has been shown to be very robust in recognizing characters written by various tools, such as pencils, ink pens, mark pen as well as the Chinese calligraphy brushes. The prototype system takes only five seconds to partition characters on an A4 size input image, and takes 270 ms in average to identify a character image out of a commonly used Chinese character set on a Pentium 100 based personal computer. For alpha-numerical character recognition, the recognition is about three times faster. Furthermore, because of the inherent parallel and distributed processing nature of BDNN, the technique can be easily implemented via specialized hardware for real-time performance.

4 Concluding Remarks

In this paper, a neural network based handwritten character recognition system is proposed and implemented on a Pentium-90 based personal computer. This system performs coarse classification, and mixture character recognition. The BDNN, a Bayesian Decision-based Neural network, is applied to implement the major modules of this system. This modular neural network deploys one subnet to take care one object (character), and therefore it is able to approximate the decision region of each class locally and precisely. This locality property is attractive especially for personal handwriting or signature identification applications. Moreover, because its discriminant function obeys probability constraint, BDNN has more nice properties such as low false acceptance/false rejection rates. On the other hand, due to the enormous number of variations involved, handwriting recognition applications still require more work before they can reach comparable performance by a human. Therefore, document analysis and recognition becomes an interesting and fascinating research topic in the field of intelligent information processing.

References

[1] P.A. Devijver and J. Kittler, *Pattern Recognition: A Statistical Approach*, Prentice-Hall, 1982.

[2] H.C. Fu and K.P. Chiang, "Recognition of Handwritten Chinese characters by Multi-stage Neural network classifiers," in *Proc. of 1995 IEEE Int. Conf. Neural Networks*, Perth, Australia.

[3] S.H. Lin, S.Y. Kung, and L.J. Lin, "A Probabilistic DBNN with applications to sensor fusion and object recognition," in *Proc. of the 5th*

IEEE workshop on Neural Networks for Signal Processing, Cambridge MA, USA, Sep., 1995, pp333-342.

[4] S. Mori, C.Y. Suen, and K. Yamamoto, "Historical Review of OCR Research and Development," *Proceedings of IEEE*, vol. 80, no. 7, pp. 1029-1058, July, 1992.

[5] L.T. Tu, Y.S. Lin, C.P. Yeh, I.S. Shyu, J.L. Wang, K.H. Joe, and W.-W. Lin, *Recognition of handprinted Chinese characters by feature matching*, in Proc. of 1991 First National Workshop on Character Recognition, Taipei, ROC, 1991, pp. 166-175.

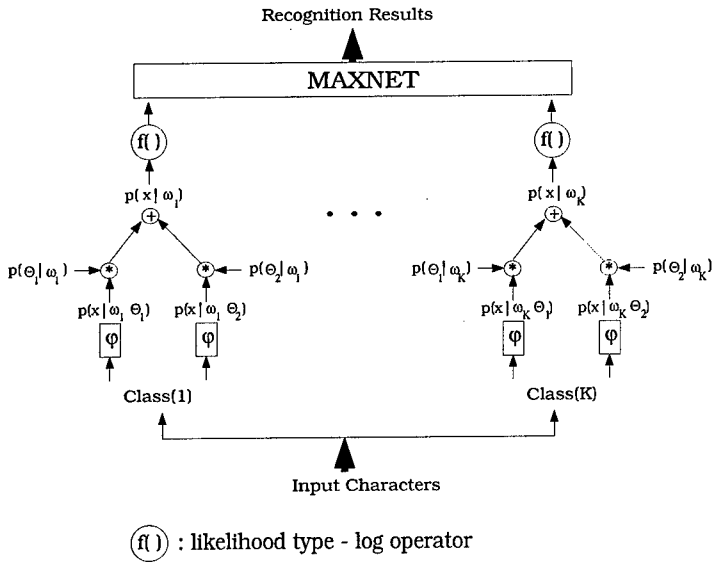


Figure 1: Architecture of a *k-class* BDNN for character recognition

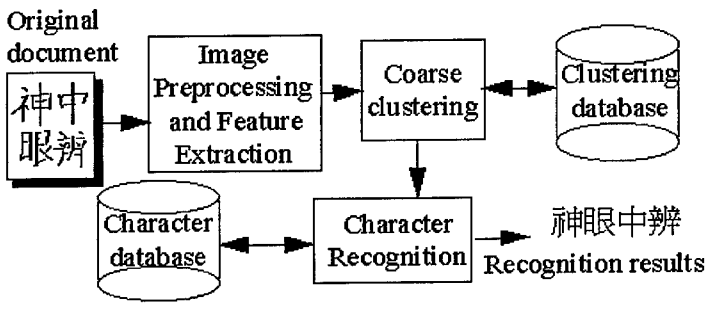


Figure 2: System configuration of the multi-stage character recognition system

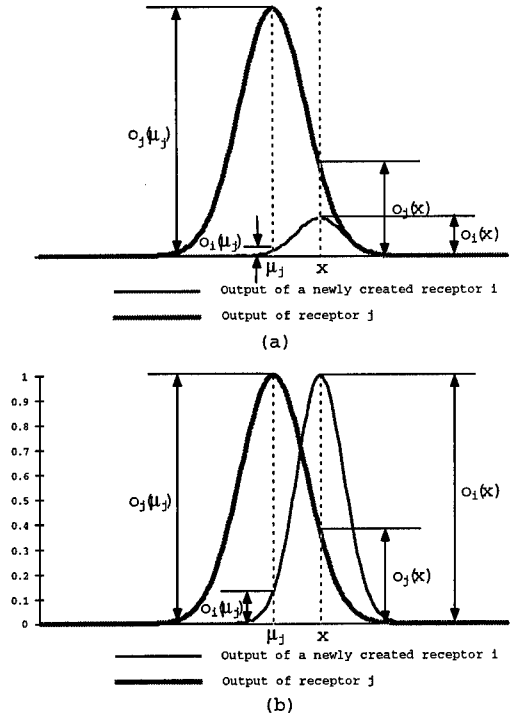


Figure 3: Example of receptive field overwhelming: (a) a newly created receptor S_i is overwhelmed by a receptor S_j; (b) the proper initialized receptor S_i and the receptor S_j.

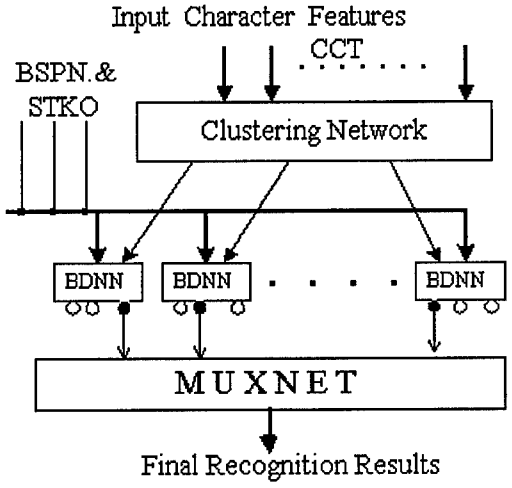


Figure 4: The architecture of the two-stage mixture character recognizer.

Neural Networks for Engine Fault Diagnostics

Dawei W. Dong*†, John J. Hopfield†, and K. P. Unnikrishnan†‡

†Computation and Neural Systems
California Institute of Technology
Mail Code 139-74
Pasadena, CA 91125

‡General Motors R&D Center, 480-106-285
Warren, MI 48090-9055

Abstract

A dynamic neural network is developed to detect soft failures of sensors and actuators in automobile engines. The network, currently implemented off-line in software, can process multi-dimensional input data in real time. The network is trained to predict one of the variables using others. It learns to use redundant information in the variables such as higher order statistics and temporal relations. The difference between the prediction and the measurement is used to distinguish a normal engine from a faulty one. Using the network, we are able to detect errors in the manifold air pressure sensor (V_s) and the exhaust gas recirculation valve (V_a) with a high degree of accuracy.

1 Introduction

The basic behavior of an automotive engine is well known (Dobner 1983, Cook and Powell 1988). In the intake manifold of an automotive engine, shown schematically in Figure 1, the mass air flow rate (V_i), exhaust gas re-circulation valve position (V_a), engine speed (V_o), and manifold absolute pressure (V_s) are related by a first order dynamics:

$$dV_s/dt = F(V_i, V_o, V_a, V_s).$$

In many automobiles, sensors directly measure the variables V_s, V_i, and V_o, and the actuator command V_a is also monitored. However, the above equation indicates that there is a redundancy between these variables. The

*To whom correspondence should be addressed. Phone: 818-395-2805. Fax: 818-792-7402. Email: dawei@hope.caltech.edu

0-7803-4256-9/97/$10.00

consistency of the time-history of the four variables can be used to check for faults in the three sensors and in the actuator. Thus for example by monitoring the variable V_s, we should be able to reliably detect errors in variables such as V_a.[†] We present a neural network model that can capture the above dynamics of a six-cylinder engine on a production vehicle. Even though the neural network presented here is for a specific engine diagnostic problem, the approach is quite general and can be easily used for other applications as well.

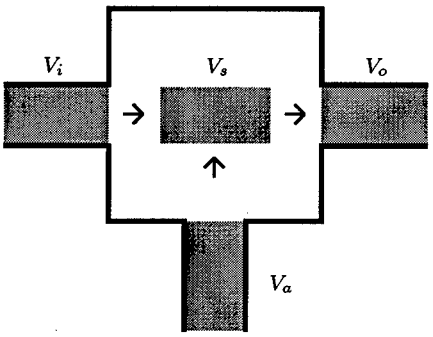


Figure 1: Engine flow diagram. There are two in-flows V_i and V_a and one out-flow V_o. Because the conservation of mass, the change of V_s, which is proportional to the total change of mass in the manifold, is proportional to the net mass flow, which is a function of V_i, V_a, V_o, and V_s.

2 Network

A two layer feedback neural network is developed to predict one variable (V_s) using three others. The architecture of the network is illustrated in Figure 2. The network has 3 feed forward inputs (V_i, V_a, and V_o), 16 first (hidden) layer neurons, and 1 second (output) layer neuron[‡]. The predicted V_s is fed back as the fourth input. The facts that (i) the first layer uses time-delayed output variables and (ii) the input-output relationship of each neuron is sigmoidal, allows the network to capture the knowledge that the physical system is characterized by a first order non-linear dynamics.

[†]There is an extensive body of literature on fault diagnosis. The inherent relationships and redundancies of measured variables of dynamic processes are often used to detect faults (e.g. Isermann 1993).

[‡]Different number of hidden neurons has been tried. 16 gives a good level of performance for this task

The data for training and testing the network was collected by a laptop computer during normal city and highway driving using an experimental hardware setup within the vehicle. The data was then loaded to a SUN workstation for training and testing. We focus on faults in the sensor V_s and the actuator V_a. The latter was chosen for its difficulty in detection. The faults were introduced by a hardware fault generator and simulates an 80% V_s fault or an 80% V_a fault. (80% V_s fault means that the sensor V_s reading is 80% of the real value. 80% V_a fault means that in the local actuator control loop for V_a, the sensor output is 80% of the actual value. This will cause the actuator V_a to open more for a given V_a command, thereby increasing the V_s by roughly 5%.)

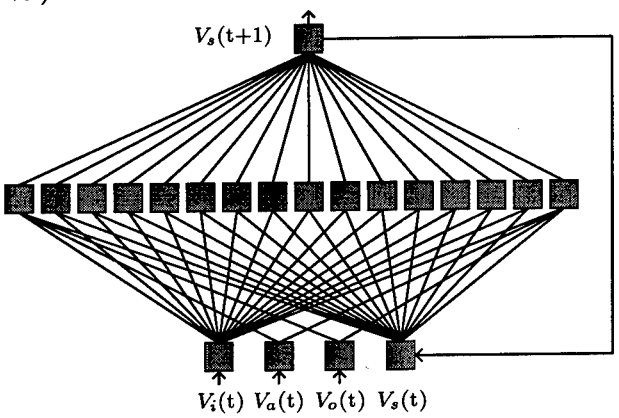


Figure 2: Network architecture. This feedback network has two layers of neurons, there are total of 80 connections (five for each hidden neuron) and 17 thresholds (one for each neuron). Those 97 parameters of the network are trained by back error propagation (BEP).

The connections of the network are trained by back error propagation (BEP) with a momentum term on a training data set. To learn the dynamic correctly, the training data are presented in the following fashion:

1) find a random starting point in a long time sequence of data, set the initial value of the feedback input to the measured V_s;
2) run the input through the network to get an output V_s, calculate the output error (the square difference of the predicted V_s and the real one);
3) set the feed forward input to the next data point and the feedback input to the predicted V_s;
4) repeat steps (2) and (3) for 100 steps to collect the error signal;

5) repeat step (1), (2), (3) and (4) for 4 steps to further collect the error signal;
6) update the connection according the BEP learning rule;
7) repeat (1) through (6) until the error does not reduce any more or until the limit of computation are reached.

The performance of the network is tested on a separate validating data set. In all the plots, except the one mentioned, the validating data set is used.

3 Network Performance

The purpose of training the close-loop (feedback) network is to facilitate the identification of a normal vehicle from a faulty one, and thus diagnose a fault. The variance (root of mean square) of the distribution of the difference between the predicted and the measured V_s is used as the quantitative measure of the network's identification power.

3.1 System Identification

The trained neural network predicts V_s value very well. This is shown in Figure 3 (left) for a segment of the normal data set, sampled every 25 ms for 400 seconds, under normal city driving conditions. The predicted V_s values (dashed line) and the measured V_s values are very close to each other.

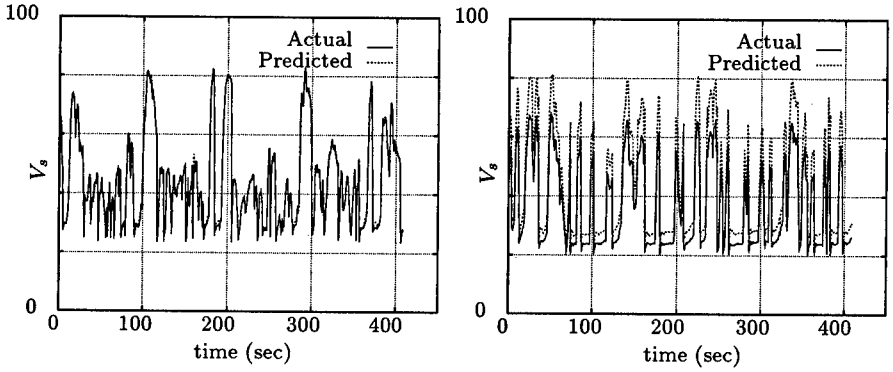


Figure 3: V_s prediction for normal vehicle (left) and faulty vehicle (right).

With a network of this accuracy, it is easy to detect faults in sensor V_s. Figure 3 (right) shows a segment of the data set which was collected with faulty V_s sensor (the reading is 80% of the true value). It is the same vehicle

but with an altered V_s sensor. We can see that the prediction is about 20% above the measured value, i.e., the network predicts a V_s value 20% greater than the actual reading, given other variables.

But if a fault only causes small changes in V_s, it is not so easy to see the difference from a plot like Figure 3. For the data set which was collected with the faulty V_a actuator (the V_a actuator opens 20% more than the V_a command), the changes in V_s is only about 5%. The difference between the predicted and measured V_s, i.e., the residual, gives a quantitative measure. The distribution of the residual is quite different for a normal vehicle and a faulty one.

3.2 Diagnostic Variable

Figure 4 (left) shows the V_s residual values for the same segment of the data set as in Figure 3 (left) for a normal vehicle. The residuals are well within 1 with the mean close to 0. Obviously there are many ways to characterize the residual, e.g., binning it for different V_s values and/or V_a values. Even the residuals at different times could give information on whether a vehicle is normal or not.

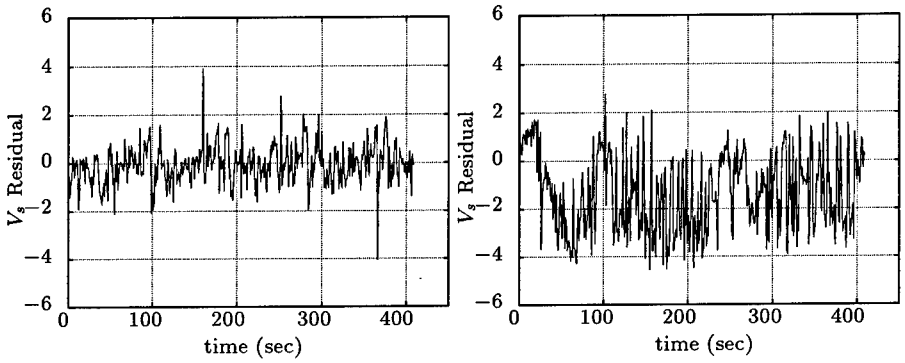


Figure 4: V_s residual for normal vehicle (left) and a vehicle with V_a fault (right).

The residual variance is the most natural one to characterize the spread of the residual distribution. For the current application, this is sufficient to separate a normal vehicle from a faulty one. For the the segment of the normal data set shown in Figure 3 (left) the running average of the residual variance is shown in Figure 5 (left, the curve in the middle). It is clear that

the residual variance converges to 0.8 within about 200 sec. This translates to roughly 1.8% of the measured V_s. We can see from Figure 5 (right) that the V_s residual variance of a vehicle with 80% V_a fault is much larger than this, so it is possible to detect the V_a fault with average V_s residual variance.

3.3 Discrimination Power

Since the V_s fault is easy to detect, only the performance of the network for detecting 80% V_a fault is presented in the following. Figure 4 (right) shows the V_s residual values for an 80% V_a fault vehicle. The V_s residuals have much larger variances, in contrast to those in Figure 4 (left) for a normal vehicle.

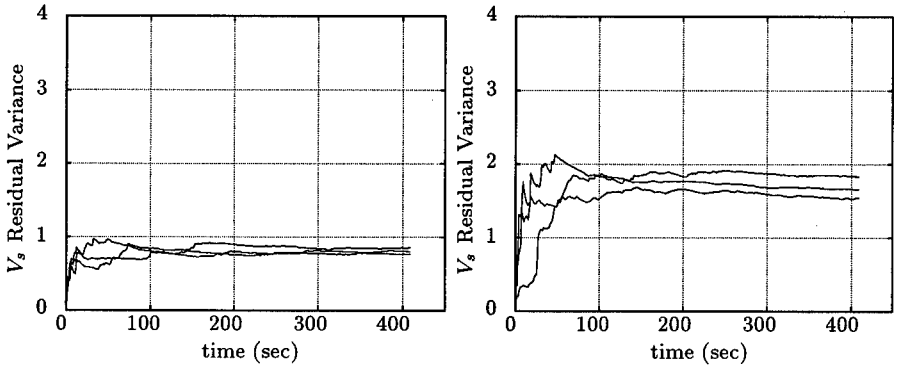


Figure 5: V_s residual variance for normal vehicle (left) and a vehicle with V_a fault (right). Three segments are plotted for each of the normal data set and the faulty data set. The variances for a faulty vehicle are larger than the normal ones by more than a factor of two.

Figure 5 shows the running average of V_s residual variances for three segments of the normal data set (left) and three segments of the V_a fault data set (right). The V_s residual variances for the faulty data set are around 1.8 , more than two times larger than 0.8 variance for the normal data set. Again, the variances approach their asymptotic values within about 200 secs. The small difference from segment to segment reflects the random driving pattern during the data collection (there was no set driving schedule).

4 Network Generalization

The most serious concern for any data dependent model (neural network and math based models alike) is how well the model generalizes. This is

investigated in two ways: variations from the training data to validating data, variations for different drivers.

4.1 Training and Validating

Figure 6 (left) shows the V_s residual variances on three segments of the training data set. These can be compared with the three segments of the validating data set in Figure 5 (left). Both the training and validating data were collected for the same driver A in this case. All the running averages of V_s residual variances approach 0.7 to 0.9 after 200 seconds. There is no significant difference in the variance for training and validating data.

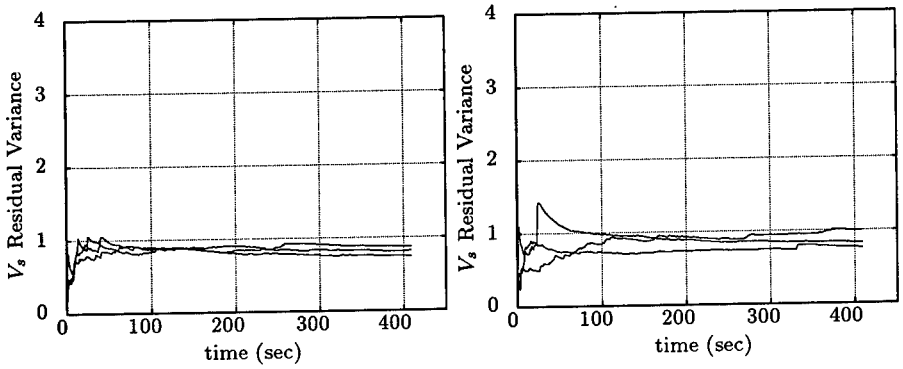


Figure 6: V_s residual variance for training (left) and for different drivers (right). On the right, the lower two curves are for driver B and the upper one is for driver C.

4.2 Different Drivers

Figure 6 (right) shows the V_s residual variances of three segments of normal data from two other drivers (B and C) which can be compared with the three segments of the validating data set as before (Figure 5, left). The network was trained for driver A. Thus the data for different drivers are not part of the training data set. The running averages of V_s residual variances for drivers B and C are only slightly higher, ranging from 0.7 to 1.1.

The small difference in performance for training, validating, and drivers is well below the level of V_s change caused by the V_a fault. Thus the network can still give a reliable signal to detect the fault.

5 Discussion

Based on the intrinsic dynamic of the intake manifold of an automotive engine (shown in Figure 1), We choose a feedback network instead of a feedforward one. To test what a feedforward network can do, we also trained a network without the V_s feedback, i.e., a standard two layer feedforward network.

Figure 7 shows the performance of the trained feedforward network. The running average of V_s residual variances for three segments of the normal data set (left) and three segments of the V_a fault data set (right) are shown in this figure. The prediction accuracy of the feedforward network is much lower than the feedback network (Figure 5). The variances for the normal data set are two times larger than the feedback network and are very close to the variances for the V_a fault data set.

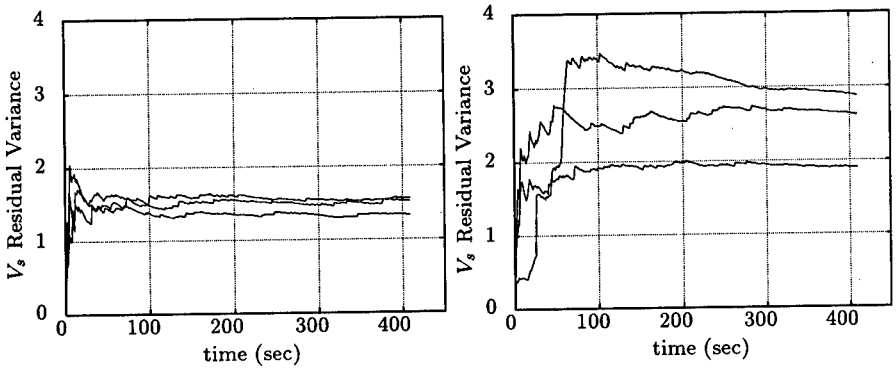


Figure 7: Performance of a feedforward network. As in Figure 5, V_s residual variance for normal vehicle is shown on the left and a vehicle with V_a fault on the right. Three segments are plotted for each of the normal data set and the faulty data set. Different from Figure 5, the variances for normal and faulty vehicles are not very far apart.

Another alternative is to train a feedforward network with true V_s, not the feedback from the output. With the same training schedule, the network learned mostly to follow the V_s input. Thus the performance is even worse than without the V_s input — in term of discriminating faulty and normal vehicles.

We have also trained feedforward networks with multiple time-delayed inputs of V_i, V_a, and V_o. They have similar level of performance as the feedforward network in Figure 7, which is much worse than the feedback one. Thus the feedback element of the network is truly important.

6 Summary

This research demonstrates the usefulness of applying neural network technology for engine modeling and diagnostics. Using a well accepted statistical measure — the variance — the two layer network with feedback achieved a accurate manifold air pressure (V_s) prediction with 1.8% variance which enables the detection of 4.5% V_s variance caused by exhaust gas recirculation valve (V_a) faults of the same vehicle.

We should point out that it is not necessary to collect the data in continuous 400 sec windows. For the current method to work, one only needs to collect small pieces of data say, 2 or 3 seconds long, and collect many pieces to accumulate enough statistics. On the other hand, collecting and processing data continuously every 25 ms itself is not very demanding. The computational needs for processing data after the network has been trained is only about 4000 multiplication per second.

Acknowledgements

We wish to thank Professor Rodney M. Goodman from Caltech for his encouragement. This project is funded by General Motors, Delco Electronics, and the NSF Engineering Research Center in Neuromorphic Systems at California Institute of Technology.

References

[1] J. A. Cook and B. K. Powell, Modeling of an internal combustion engine for control analysis. IEEE Control Systems Magazine. pp 20-26 (1988).

[2] D. J. Dobner, Dynamic engine models for control development — part I: non-linear and linear model formulation. Int. J. of Vehicle Design. Special Publication SP4 pp 54-74 (1983).

[3] R. Isermann, Fault diagnosis of machines via parameter estimation and knowledge processing — tutorial paper. Automatica. Vol 29(4) pp 815-835 (1993).

LOCALLY RECURRENT NETWORKS WITH MULTIPLE TIME-SCALES

Jui-Kuo Juan, John G. Harris and Jose C. Principe
Computational Neuro-Engineering Laboratory
University of Florida
453 NEB Bldg, Gainesville, FL 32611
Email: harris@cnel.ufl.edu

Abstract - We introduce a new generalized feed-forward structure that provides for multiple time scales. The gamma, Laguerre and other locally recurrent feed-forward structures perform poorly in cases where widely varying time constants are required. By exponentially varying the time-constant along the delay line, a single delay line is able to represent signals that include various time scales. We demonstrate both discrete- and continuous-time versions of this multiple time-scale structure which we call the multi-scale gamma filter. The multi-scale gamma has a very natural implementation in sub-threshold CMOS and measured impulse responses from a continuous-time analog VLSI chip are shown.

1 INTRODUCTION

For many practical problems, the gamma structure is superior to the standard tap delay line because of its ability to automatically choose an appropriate time-scale [1] [2] [3]. This advantage becomes particularly significant for problems involving extremely long impulse responses for which the standard tap delay line solutions can require thousands of taps. Unfortunately, the gamma structure has a few problems of its own:

1. Choosing the optimal time scale is a very difficult computational problem. Gradient descent is not guaranteed to find the optimal time scale[4]. This problem becomes particularly troublesome when we build dedicated hardware (analog or digital) for implementing these filters.

2. Even when a single optimal time-scale can be found, the structure may not be able to efficiently represent information occurring at other time scales.

0-7803-4256-9/97/$10.00

3. As with the FIR case, choosing an appropriate number of taps is also a difficult optimization procedure.

Sections II and III discuss the discrete- and continuous-time realizations of a multiple time-scale generalization of the gamma filter which addresses the above problems. Section IV discusses an analog VLSI hardware implementation of this filter.

2 DISCRETE-TIME MULTI-SCALE GAMMA FILTER

Our proposed discrete-time multi-scale gamma filter is shown in Figure 1. Unlike the gamma filter, the location of the pole at each stage depends on the tap k. The transfer function between taps can be written as:

$$\frac{x_k(z)}{x_{k-1}(z)} = \frac{a^{k-1} \cdot \mu}{z - (1 - a^{k-1} \cdot \mu)} \tag{1}$$

(We assume that $a < 1$.) If we perform a two-dimensional search for the optimal a and μ, this structure cannot perform worse than the original gamma, since the gamma filter is included as a special case when $a = 1$. We can

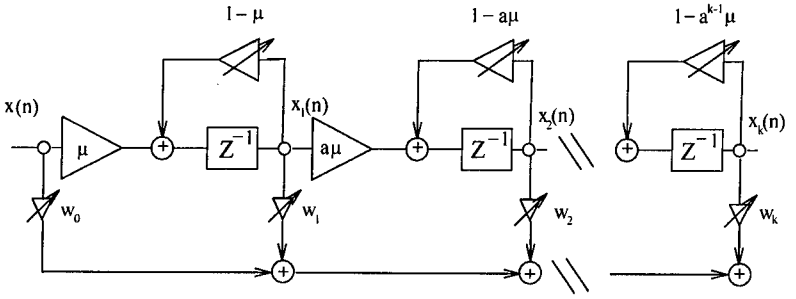


Figure 1: *Discrete-time multi-scale gamma filter*

demonstrate a problem for which the multi-scale filter easily outperforms the gamma filter by posing a system ID problem that includes two widely separated poles. We use the transfer function H(z):

$$H(z) = \frac{0.3}{(z - 0.05)(z - 0.95)} \tag{2}$$

For the gamma structure, we scan all possible values of μ between 0 and 1. For convenience, we set $\mu = 1$ in the multi-scale gamma so that the first stage is exactly an ideal delay. We then scan all values of of a between 0 and 1. The Wiener-Hopf equations were used to solve for the optimal weight

values in order to obtain the MSE. The results for various numbers of taps (from 2 to 5) are shown in Figure 2. In all cases, the gamma filter has more difficulty in approximating the system with a small number of taps. Since we are scanning all values of the free parameter, a practical optimization procedure is still an open problem. The solution for the multi-scale gamma can be further improved if we also optimize for μ instead of setting $\mu = 1$. Currently we are not exploring this direction because we seek simple search methods that can be readily implemented in dedicated hardware. The multi-

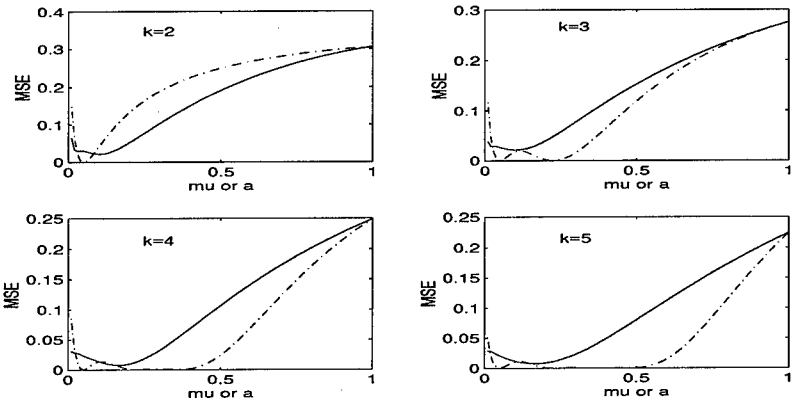


Figure 2: *Mean square error comparisons between the gamma filter (solid line) and the multi-scale gamma filter (dotted line). We plot MSE vs μ (for gamma) or a for (multi-scale gamma).*

scale gamma structure is now better able to represent signals with widely varying time constants but we still require a difficult optimization procedure to find the optimal a (assuming $\mu = 1$). Rather than perform this difficult search procedure, we borrow a standard weight pruning technique from neural network theory [5]. We purposely include more taps than we need in order to cover a large range of time scales and selectively deactivate any weight values that are small in magnitude.

We have simulated a system ID problem using the multi-scale gamma filter in which the transfer function of the unknown system is given by:

$$H(z) = \frac{0.4(z-1)}{z(z-0.1)(z-0.7)(z-0.9)} \tag{3}$$

The Mean Square Error vs. number of taps is shown in Figure 3. For an n-tap filter, the smallest $10-n$ weight magnitudes are set to zero. We again assume that $\mu = 1$. There are a few things to note in Figure 3. First, the $a = 0.9$ solution is better than $a = 0.5$ for an equal number of taps. For $a = 0.5$, the poles are spaced too far apart. Second, for both values of a there is a sharp transition beyond which adding more taps does not decrease the error. This sharp transition can be used to choose a reasonable

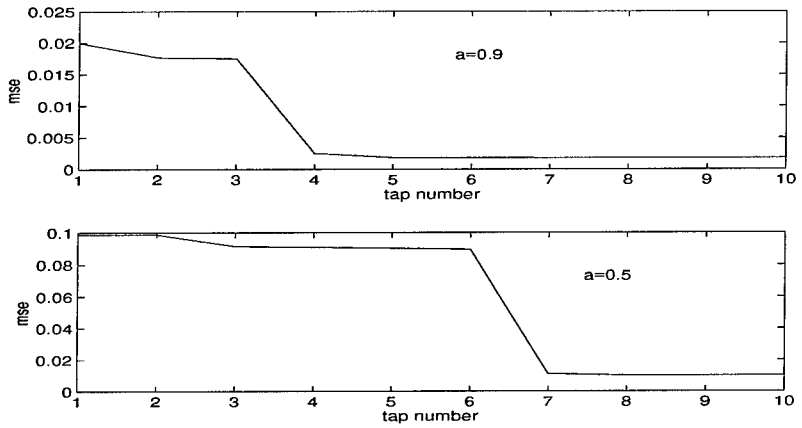


Figure 3: *Plot of MSE vs. number of taps using the weight pruning method. For each number of taps, the smallest magnitude weights are set to zero.*

number of taps for each problem. Minimizing the number of taps reduces the overall amount of computation and also lowers the misadjustment of the system. We plan to use this weight pruning method to avoid implementing complex, non-convex optimization procedures in our dedicated hardware. The weight pruning method also provides a mechanism for choosing an appropriate number of taps.

3 CONTINUOUS-TIME MULTI-SCALE GAMMA FILTER

Our continuous-time multi-scale gamma (ms-gamma) structure is shown in Figure 4. The ms-gamma is a cascade of first-order low-pass filters with time constants that slow down exponentially as signals propagate down the cascade. If we define the time constant of the last stage to be τ, then the next to the last stage has a time constant of $a\tau$ where $0 < a < 1$. Since

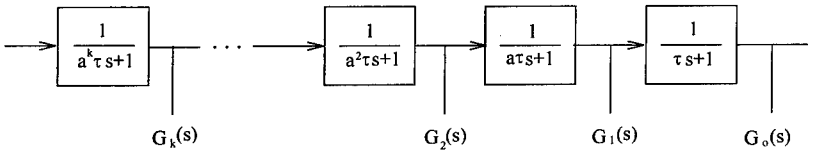


Figure 4: *Continuous-time multi-scale gamma memory*

the time constant changes by a factor of a for each stage, if we set $a = 1$, the ms-gamma reduces to the usual gamma memory. We can simplify the mathematical analysis by considering an infinite cascade of sections. In general, for $a < 1$

$$H_k(s) = \frac{1}{a^k \tau s + 1} \tag{4}$$

And full system response at tap k is given by

$$G_i(s) = \prod_{k=i}^{\infty} \frac{1}{a^k \tau s + 1} \tag{5}$$

Because of the infinite cascade, the following scaling laws can be easily derived. In the s-domain:

$$G_{i+1}(s) = G_i(as) \tag{6}$$

In the time-domain:

$$g_{i+1}(t) = \frac{1}{a} g_i(\frac{t}{a}) \tag{7}$$

Therefore the impulse functions $g_i(t)$ are all identical with the exception of a scaling of the amplitude and time axis.

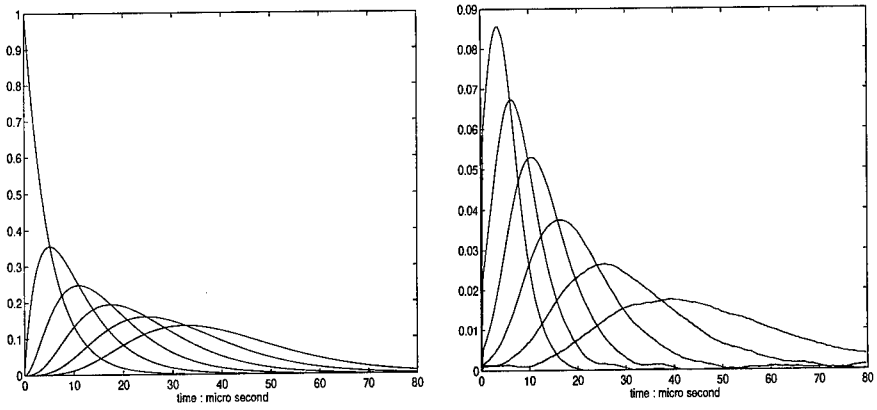
We have derived an analytic form of the impulse response at each tap for the ms-gamma for both the finite and infinite cascade versions. Both expressions consist of a weighted sum of exponentials. The expression of the impulse response of the infinite cascade is simpler and can be written as:

$$g_i(t) = \sum_{k=i}^{\infty} \frac{A_{k-i}}{a^k \tau} e^{\frac{-t}{a^k \tau}} \tag{8}$$

where

$$A_m = \prod_{\substack{j=0 \\ j \neq m}}^{\infty} \frac{1}{1 - a^{j-m}} \tag{9}$$

Figure 5 shows the impulse response curves of ms-gamma both simulated and measured. The peak values of the impulse response are equally spaced



(a) (b)

Figure 5: *(a) Simulated multi-scale gamma kernels. (b) Measured kernels.*

for the gamma filter but are not equally spaced for the ms-gamma. For the infinite cascade, define t_i to be the peak value of the impulse response $g_i(t)$, the scaling law results in the following relation:

$$t_{i+1} = at_i \tag{10}$$

which means that the time of the peak at stage $i+1$ is simply the product a and the time of the peak at stage i. This implies that the peak value of the impulse response for consecutive taps of the infinite cascade are equally spaced on a log time plot. Figure 6 shows the peak location of 10 normalized tap responses on a log time plot for the finite ms-gamma memory. Notice that after the first few taps, the peak values become equally spaced and the impulse response shapes converge to the same shape. This is exactly what is expected from the infinite cascade analysis.

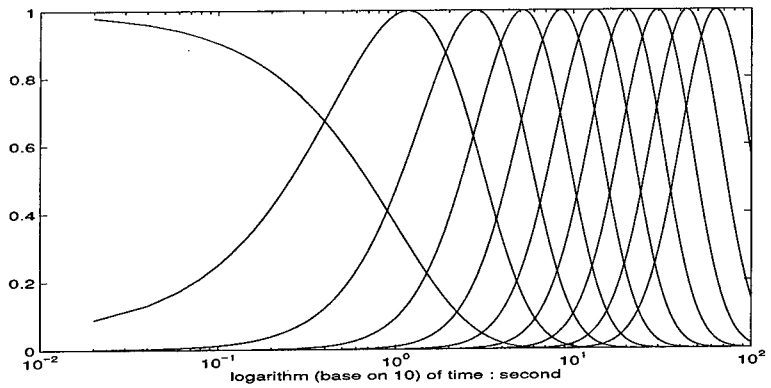


Figure 6: *Impulse responses of the multi-scale gamma memory on a log-time plot. The peak value of each impulse response has been normalized to unity for display purposes.*

We have also performed system identification simulations using the ms-gamma filter in continuous-time. Figure 7 shows that the ms-gamma performance index is fairly flat when $\frac{1}{\tau}$ is larger than 800. Since the performance surface is fairly constant and close to its optimal value in this region, finding the optimal τ may not be so important for this structure. Figure 8 shows that when the number of taps (k) increases, the performance index becomes even more flat. With our example, the unknown system is a 5^{th} order system and using a 5^{th} order ms-gamma filter is enough to approximate the unknown system. Increasing the number of taps to $k = 6$ does not provide much improvement. These results suggest that there may be no need to find an optimal scale (as is necessary for the gamma) if many time scales are explored simultaneously (as in the ms-gamma). The misadjustment of the system may be reduced by systematically zeroing out any weights that do not contribute significantly to the output.

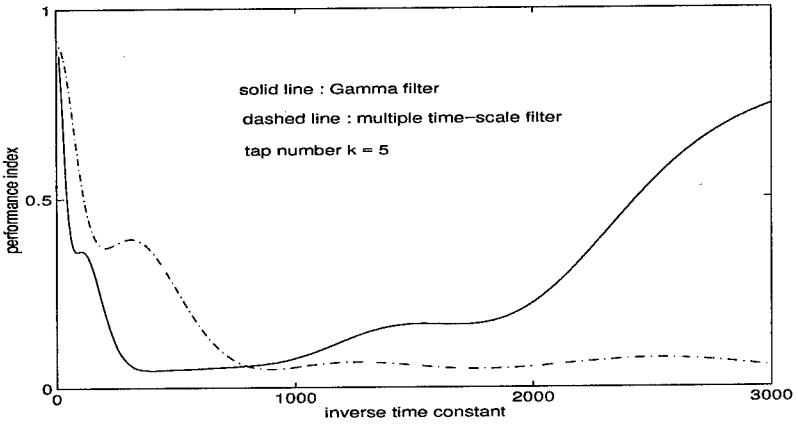


Figure 7: *Performance index comparison of gamma and multi-scale gamma structures.*

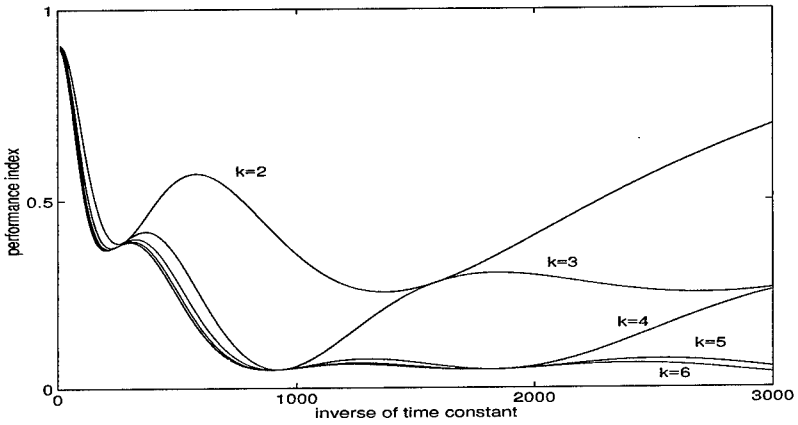


Figure 8: *Performance index of multi-scale gamma with different numbers of taps (k).*

4 ANALOG HARDWARE IMPLEMENTATION

We have previously implemented the gamma filter in analog VLSI [6] and have integrated a continuous-time LMS gradient decent method to determine the weight values [7]. In order to implement the multi-scale filter, a resistive line is connected along the tap bias controls to achieve a linear voltage drop from one end to the other. Because the CMOS transamps are operated in the sub-threshold region, the output currents are exponential in the bias voltages, which means their poles also exponential decreasing as k increases. A similar strategy was used in the implementation of the silicon cochlea [8]. Figure 9 shows the schematic for the k-tap multi-scale gamma structure. The resistors are chosen with the same value, the time constant τ is controlled by the

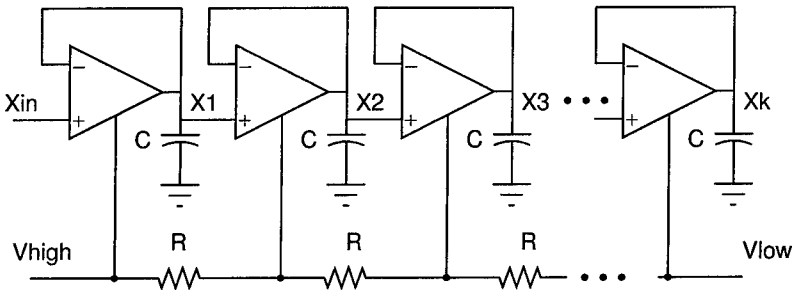


Figure 9: *Circuit implementation of the multi-scale gamma structure.*

voltage V_{high} and factor a is set by V_{low}. A twelve-tap multi-scale gamma structure had been fabricated using MOSIS 2μm N-WELL technology. The measured impulse responses from the chip are shown in Figure 5(b). This analog implementation provides a fast, low-cost, low-power solution for many adaptive filtering applications.

5 CONCLUSION

We have introduced the multi-scale gamma to allow for widely varying time-scales in the input signals. The gamma, Laguerre and other locally recurrent feedforward structures perform poorly in cases where widely varying time constants are required. By exponentially varying the time constant along the delay line, a single delay line is able to represent signals that include various time scales. Our results also suggest that we may be able to skip the difficult search procedures required to find a single optimal time constant as is necessary for the standard gamma and Laguerre filters. We demonstrate both discrete- and continuous-time versions of the multi-scale gamma structure. The same extension can be applied to the Laguerre memory and most other locally recurrent networks. These multiple time-scale structures have a natural implementation is sub-threshold CMOS and measured impulse responses from a continuous-time multi-scale gamma chip were shown.

Acknowledgments

This work was supported by an Office of Naval Research contract #N00014-94-1-0858 and an NSF CAREER award #MIP-9502307.

REFERENCES

[1] J. Kuo and J. Principe. Noise reduction in state space using the focused gamma model. *Proc. ICASSP94*, vol 2:533–536, 1994.

[2] J. Principe, A. Rathie, and J. Kuo. Prediction of chaotic time series with neural networks. *Nonlinear dynamic of the brain(Editor B. Jansen)*, pages 250–258, 1993.

[3] J. Principe and M. Motter. System identification with dynamic neural networks. *Proc. W. Cong. Neural Networks*, vol 2:284–289, 1994.

[4] J. Principe and B. Vries. The Gamma filter-a new class of adaptive IIR filters with restricted feedback. *IEEE Transactions on Signal Processing*, 41(2):649–656, February 1993.

[5] S. Haykin. *Neural Networks: A Comprehensive Foundation.* IEEE Computer Society Press, 1994.

[6] J. Juan, J.G. Harris, and J.C. Principe. Analog VLSI implementations of continuous-time memory structures. *Proceedings of IEEE Int. Symp. Circuits and Systems*, 3:338–340, 1996.

[7] J. Juan, J.G. Harris, and J.C. Principe. Analog hardware implementations of adaptive filter structures. In *to appear in the Proceedings of International Conference on Neural Networks*, 1997.

[8] C. Mead. *Analog VLSI and Neural Systems.* Addison-Wesley, 1989.

MEDICAL IMAGE ANALYSIS BY PROBABILISTIC MODULAR NEURAL NETWORKS

Yue Wang[1] Tülay Adalı[2] Sun-Yuan Kung[3]
[1]Department of Electrical Engineering
The Catholic University of America, Washington, DC 20064
[2]Department of Computer Science and Electrical Engineering
University of Maryland Baltimore County, MD 21228
[3]Department of Electrical Engineering
Princeton University, Princeton, NJ 08544

A probabilistic neural network based technique is presented for unsupervised quantification and segmentation of the brain tissues from magnetic resonance image. The problem is formulated as distribution learning and relaxation labeling that may be particularly useful in quantifying and segmenting abnormal brain tissues where the distribution of each tissue type heavily overlaps. The new technique utilizes suitable statistical models for both the pixel and context images. The quantification is achieved by model-histogram fitting of probabilistic self-organizing mixtures and the segmentation by global consistency labeling through a probabilistic constraint relaxation network. Experimental results show the efficient and robust performance of the new algorithm.

1. INTRODUCTION

Quantitative analysis of brain tissues refers to the problem of estimating tissue quantities from a given image and segmentation of the image into contiguous regions of interest to describe the anatomical structures. The problem has recently received much attention largely due to the improved fidelity and resolution of medical imaging systems, and because of its ability to deliver high resolution and contrast, magnetic resonance (MR) imaging has been the dominant modality for research on this problem. For quantification of brain tissues from MR images, stochastic model based approach has been by far the most popular. The stochastic model based approach typically employs a finite mixture model, which we have shown in our recent study of MR image statistics, is a very suitable model for the task. Therefore, probabilistic neural networks are particularly suitable for application in quantitative analysis of MR images, since while providing a formal statistical formalization of the problem they also offer efficient on-line computation of the quantities of interest, a feature especially important for evaluation of studies in a clinical setting, for example an analysis to be performed on a sequence of MR images.

0-7803-4256-9/97/$10.00

In this paper, we present a probabilistic neural network approach for efficient analysis of brain tissues by using single-valued MR brain scans. The procedure provides a complete treatment of the problem of quantitative image analysis in that, a quantification stage that includes automatic determination of tissue types is followed by a segmentation stage which incorporates local spatial context with global statistical description of pixel intensities for reliable description of the anatomical structures. In particular, we formulate tissue quantification as a distribution learning problem and use relative entropy as the information distance between the standard finite normal mixture (SFNM) distribution and the image histogram. The actual quantification is performed by a probabilistic self-organizing mixtures (PSOM) network in which we use an information theoretic criterion, the minimum conditional bias/variance (MCBV) criterion, to determine the suitable number of mixture components in a given MR image. The procedure is fully unsupervised so it is able to work with both normal and abnormal cases. The actual segmentation is performed, after quantification, through combining maximum likelihood thresholding and stochastic regularization, by a probabilistic constraint relaxation network (PCRN). Experimental results demonstrate the efficient and reliable performance of the proposed scheme, in terms of the quantification achieved by PSOM, consistency of the order determination by using the proposed information-theoretic criterion, MCBV, and the final segmentation results by PCRN.

2. PROBLEM STATEMENT

Over the last few years, considerable success has been reported in MR image analysis both by using finite mixture distributions and by neural networks based methods. And very recently, a cross fertilization of these two approaches, probabilistic neural networks have emerged as a powerful tool in MR image analysis such as tissue quantification and segmentation. New approach provides valuable insight for designing and learning in neural networks, such as consistency of parameter estimates and determination of suitable network structure among others.

Assume that the spatial location of each pixel x_i, has one-to-one correspondence to its true label l_i^*. By randomly reordering all pixels in the underlying probability space, i.e., ignoring information regarding the spatial ordering of pixels, we can treat pixel labels as random variables and introduce a probability measure by using a multinomial distribution with unknown parameters π_k for each component. Since it reflects the distribution of the number of pixels in each component, π_k can be interpreted as a prior probability of the global context information. Thus, the relevant (sufficient) statistics are the tone statistics for each component and the number of pixels in each of the component. The marginal probability measure for any pixel image, i.e., the SFNM distribution, can be obtained by writing

the joint probability density of x_i and l_i and then summing the joint density over all possible outcomes of l_i, resulting in a sum of the following general form:

$$f(u|\mathbf{r}) = \sum_{k=1}^{K} \pi_k g(u|\mu_k, \sigma_k^2) \tag{1}$$

with $\sum_{k=1}^{K} \pi_k = 1$ and

$$g(u|\mu_k, \sigma_k^2) = \frac{1}{\sqrt{2\pi}\sigma_k} \exp(-\frac{(u-\mu_k)^2}{2\sigma_k^2})$$

where μ_k and σ_k^2 are the mean and variance of the kth Gaussian kernel. We use K to denote the number of Gaussian components and $\mathbf{r} \in \mathcal{R}^{3K-1}$ to denote the total parameter vector that includes μ_k, σ_k^2, and π_k for all K components.

On the other hand, since in tissue segmentation, context information is of particular importance, by assuming that the context images are random variables with Markovian property, a localized SFNM distribution can be formulated to incorporate local regularities statistically, i.e., to impose local consistency constraints on context images in terms of a stochastic regularization scheme. For each pixel i, we define the spatial constraint as a local set of all pairs (l_i, l_j) such that the consistency between l_i and l_j can be measured by the compatibility function $I(l_i, l_j)$. We define $I(\cdot,\cdot)$ as the indicator function and define the neighborhood of pixel i, ∂i by opening a $b \times b$ window with pixel i being the central pixel where b is assumed to be an odd integer. Note that pairs of labels are either compatible or incompatible in this case. Then, we compute the frequency of neighbors of pixel i with labels compatible to an assumed label of pixel i, denoted by $\pi_k^{(i)}$, given the labels of its neighbors $\mathbf{l}_{\partial i} \in \mathcal{R}^{b^2-1}$ by

$$\pi_k^{(i)} = p(l_i = k|\mathbf{l}_{\partial i}) = \frac{1}{b^2-1} \sum_{j \in \partial i} I(l_i = k, l_{j|\partial i}) \tag{2}$$

and the localized SFNM distribution for x_i directly follows by

$$q(x_i|\mathbf{l}_{\partial i}) = \sum_{k=1}^{K} \pi_k^{(i)} \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left(-\frac{(x_i-\mu_k)^2}{2\sigma_k^2}\right) \tag{3}$$

where $\pi_k^{(i)}$ is interpreted as the *conditional* prior of component determined by the uncertainty introduced by $\mathbf{l}_{\partial i}$.

Tissue quantification addresses the combined estimation of regional component parameters $(\pi_k, \mu_k, \sigma_k^2)$ and the detection of the structural parameter K in Eq. (1) given the pixel images $\mathbf{x}$. A distance minimization approach is developed where the mixture density is fitted to the histogram

of the data by finding the optimal parameters with respect to a distance measure. We use relative entropy (the Kullback-Leibler distance) [?] for tissue quantification in MR images. Relative entropy measures the information theoretic distance between the "true" distribution $f_{\mathbf{x}}(u)$ and the estimated SFNM distribution $f(u|\mathbf{r})$, and is given by

$$D(f_{\mathbf{x}}||f_{\mathbf{r}}) = \sum_{u} f_{\mathbf{x}}(u) \log \frac{f_{\mathbf{x}}(u)}{f(u|\mathbf{r})}. \tag{4}$$

Note that the use of the relative entropy cost also overcomes problems such as convergence at the wrong extreme faced by the squared error cost function as it weighs errors more heavily when probabilities are near zero and one, and diverges in the case of convergence at the wrong extreme [4]. We have shown that, when relative entropy is used as the distance measure, distance minimization is equivalent to maximum likelihood (ML) estimation of the SFNM parameters.

Anatomical structure, in addition to the the results of tissue quantification that reveals different tissue properties, provides very valuable information in medical applications. Tissue segmentation is a technique for partitioning the image into meaningful regions corresponding to the objects. Tissue segmentation may be considered as a clustering process where the pixels are classified into the attributed tissue types according to their gray-level values and spatial correlation. A reasonable assumption is the spatially close pixels are likely to belong to the same tissue type. Accordingly, tissue segmentation addresses the realization of context images l_i, $i = 1, \cdots, N$, given the observed pixel images $\mathbf{x}$. Based on the localized SFNM formulation (3), a deterministic relaxation labeling can be used to update the context images after global tissue quantification by locally minimizing the pixel classification error. With a motivation similar to the one in [2, 3], the general technique seeks for a consistent labeling solution where the criterion is to maximize global consistency measure by using a system of inequalities. The structure of relaxation labeling is motivated by two basic considerations: 1) decomposition of a global computation scheme into a network performing simple local computations; 2) suitable use of local context regularities in resolving ambiguities.

3. METHOD AND ALGORITHMS

In this paper, we present the theory and algorithms for the two stages; (1) quantification which involves network order selection and adaptive computation of the parameters to achieve both classification, and (2) segmentation which uses the order and the parameters computed in the quantification stage to perform hard classification by incorporating local context constraints.

Since the prior knowledge on the true structure of a real image is generally unknown, it is most often desirable to have a neural network structure that is adaptive, in the sense that the number of local components is not fixed beforehand. In the probabilistic neural network scheme we propose for MR image analysis, using a smaller or larger number of mixture components than the number of tissue types represented on a particular slice will result in incorrect identification and quantification of the tissues in that particular slice. This situation is particularly critical in real clinical application where the structure of the individual slice for a particular patient may be arbitrarily complex. We proposed a new information theoretic criterion formulation, the MCBV criterion, to solve the model selection problem. Our approach has a simple optimal appeal in that it selects a minimum conditional bias and variance model, i.e., if two models are about equally likely, MCBV selects the one whose parameters can be estimated with the smallest variance. A practical MCBV formulation with code-length expression is further given by

$$\mathrm{MCBV}(K) = -\log(\mathcal{L}(\mathbf{x}|\hat{\mathbf{r}}_{ML})) + \sum_{k=1}^{K_a} \frac{1}{2} \log 2\pi e Var(\hat{\mathbf{r}}_{kML}) \qquad (5)$$

where $\mathcal{L}(\cdot)$ denotes the joint likelihood function and $\hat{\mathbf{r}}_{kML}$ is the maximum likelihood estimate.

Recently, on-line versions of the EM algorithm are proposed for large scale sequential learning in maximum likelihood estimation. Such a procedure obviates the need to store all the incoming observations, changes the parameters immediately after each data point allowing for high data rates. The PSOM we present here is a fully unsupervised and incremental stochastic learning algorithm. The scheme provides winner-takes-in probability (Bayesian "soft") splits of the data, hence allowing the data to contribute simultaneously to multiple tissues. By adopting a stochastic gradient descent scheme for minimizing $D(f_{\mathbf{x}}||f_{\mathbf{r}})$, the corresponding on-line formulation is obtained by

$$\mu_k{}^{(t+1)} = \mu_k^{(t)} + a(t)(x_{t+1} - \mu_k^{(t)}) z_{(t+1)k}^{(t)}, \quad k = 1, ..., K. \qquad (6)$$

$$\sigma_k^{2(t+1)} = \sigma_k^{2(t)} + b(t)[(x_{t+1} - \mu_k^{(t)})^2 - \sigma_k^{2(t)}] z_{(t+1)k}^{(t)}, \quad k = 1, ..., K. \qquad (7)$$

$$\pi_k^{(t+1)} = \frac{t}{t+1} \pi_k^{(t)} + \frac{1}{t+1} z_{(t+1)k}^{(t)}. \qquad (8)$$

where the variance factors are incorporated into the learning rates while the posterior Bayesian probabilities are kept, and $a(t)$ and $b(t)$ are introduced as the learning rates, two sequences converging to zero, ensuring unbiased estimates after convergence. Self organization at both the neuron and modular levels refers to a specific human brain capability, which tends to convert the similarity of input features into the proximity of finite

participating neurons [5, 10]. Mapping this operation to the PSOM, we design a network where both the structure and weights are updated according to an unsupervised learning algorithm. More precisely, the network organizes itself to efficiently map the data to the feature space through adaptive mechanisms where the information theoretic criteria are shown to provide a reasonable approach for the solution of the problem. In particular, both structure and weights of the PSOM "compete" for the assignment order of each model and assignment probability of each observation. Overall convergence dynamics of the PSOM are similar to SOM in that a solution is obtained by "resonating" between input data and an internal representation. Such a mechanism can be considered as a more realistic learning than the batch EM procedure.

Given the SFNM parameters, i.e. the image components computed by ML principle, there are several approaches to perform pixel classification. When the true pixel labels l_i^* are considered to be functionally independent and non-random constants, competitive learning approaches can be used for the segmentation of different tissue types. We can define the consistency of discrete relaxation labeling and formalize its relationship to global optimization as follows: We first define the component in the localized SFNM distribution (3) as a support function:

$$S_i(k) = \pi_k^{(i)} \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left(-\frac{(x_i - \mu_k)^2}{2\sigma_k^2}\right). \tag{9}$$

Note that the support function $S_i(k)$ is a function of the component (tissue type) k. Then tissue segmentation is interpreted as the satisfaction of a system of inequalities:

$$S_i(l_i) \geq S_i(k), \tag{10}$$

for all k and for $i = 1, \cdots, N$, where a consistent labeling is defined as the one having maximum support at each pixel simultaneously. We further define the average local consistency measure

$$A(\mathbf{l}) = \sum_{i=1}^{N} \sum_{k} I(l_i, k) S_i(k) \tag{11}$$

to link consistent labeling to global optimization. It is shown that when the spatial compatibility measure is symmetric and $A(\mathbf{l})$ attains a local maximum at $\mathbf{l}$, then $\mathbf{l}$ is a consistent labeling [2, 6, 8]. Hence, a consistent labeling can be accomplished by locally maximizing $A(\mathbf{l})$. We propose a probabilistic constraint relaxation network (PCRN) to perform contextual tissue segmentation by imposing neighborhood context regularities to alleviate the ambiguity problem. PCRN uses stochastic discrete gradient descent procedure where each pixel is randomly visited and its label is updated, i.e., pixel i is classified into the kth region if

$$l_i = \arg\left\{\min_k \left(\log(\sigma_k^2) - 2\log(\pi_k^{(i)}) + (x_i - \mu_k)^2 \sigma_k^{-2}\right)\right\} \tag{12}$$

where $\pi_k^{(i)}$ is defined in (2).

Rather than minimizing an energy function but looking for a possible local maximum of a global consistency measure, in PCRN the input layer has a neuron that corresponds to each pixel image and the output layer has a neuron that corresponds to the label of the original image. Competition within hidden layer ensures that only one neuron becomes active at any pixel location. Gating between output and the hidden layer incorporates the local labeling information to provide locally consistent labeling and hence to remove the ambiguities. Reciprocal feedback from output to gating unit allows each hidden neuron to control its activation. We can view consistency as a "locking-in" property, i.e., since the support function defined for a given pixel depends on the current labels at neighboring pixels, this neighborhood influences the update of the given pixel through probabilistic compatibility constraints. With the constraint propagation, the relaxation process iteratively updates the label assignments to increase the consistency, find a more consistent labeling with the neighboring labels, ideally so that each pixel is designated a unique label [2].

4. EXPERIENTAL RESULTS AND DISCUSSIONS

In this section, we present results using the probabilistic neural network based approach we introduced to quantify and segment tissue types from real MR brain images. We present a simulation study to test the performance of model identification (selection and quantification) with the proposed criterion (MCBV). We generate a test data with up of four overlapping normal components. Each component represents one local cluster. The value for each component is set to a constant value and normal distributed noise is then added to the data. The phantom, the MCBV curve as a function of the number of local clusters K, and the final distribution learning, are plotted in Figure 1. According to the information theoretic criteria, the minima of the curve indicate the correct number of the image components. The result shows that the number of local clusters suggested by the new criterion is correct and the histogram-model fitting is satisfactory.

For the real MR brain image, information theoretic criterion is first applied to detect the number of tissue types thus allowing the corresponding network to adapt its structure for the best representation of the data. The PSOM algorithm is used to quantify the parameters of the tissue types leading to a ML estimation. Segmentation of identified tissue components is then implemented by PCRN through contextual Bayesian decision. Figure 2 (a) shows the original data consisting of pure brain tissues, T1-weighted image parallel to the AC-PC line, acquired with a GE Sigma 1.5 Tesla system. The imaging parameters are TR 35, TE 5, flip angle 45^0, 1.5 mm effective slice thickness. The corresponding histogram is given in Figure 2

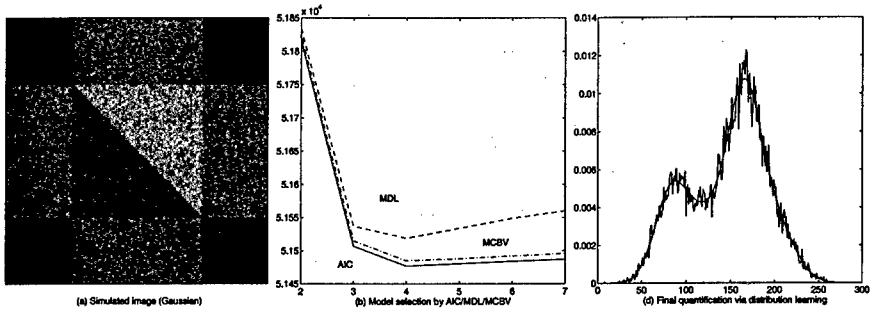


Figure 1: Experimental results of model selection and final quantification on the simulated image.

(b). that has a considerably complex characteristics since the tissue types are all highly overlapping. Evaluation of different image analysis techniques is a particularly difficult task and dependability of evaluations by simple mathematical measures is largely in question. The quality of the quantified and segmented image usually depends heavily on the subjective and qualitative judgements. In this study, besides the evaluation performed by radiologists, we use the global relative entropy (GRE) value to reflect the quality of tissue quantification and for assessment of tissue segmentation, we use post-segmentation sample averages as an indirect but objective criterion. As discussed in the literature, the brain is generally composed of three principal tissue types, i.e., WM, GM, CSF, and their pair-wise combinations, called partial volume effect. Since the MRI scans clearly show the distinctive intensities at the local barin areas, the functional tissue types need to be considered. We let $K_{min} = 2$ and $K_{max} = 9$ and calculate MCBV(K) (5) ($K = K_{min}, ..., K_{max}$). The result suggested that the brain image contains 8 tissue types. When performing the computation of the information theoretic criteria, we used PSOM to iteratively quantify different tissue types for each fixed K. The results of final tissue quantification with $K_0 = 8$ is shown in Figure 2 (b) where a GRE value of 0.02 - 0.04 nats is achieved. The PCRN tissue segmentation is performed where PCRN updates are terminated after 5-10 iterations since further iterations produced almost identical results. The segmentation result is shown in Figure 2 (c). Although the segmentation contains some small isolated spots (less than 4-pixel size), the PCRN approach is quite encouraging. These quantified tissue types agree with that of a physician's qualitative analysis results.

We also present a comparison of the performance of PSOM with that of the EM and the competitive learning (CL) algorithms in MR brain tissue quantification, to evaluate the computational accuracy and efficiency of the algorithm in the standard finite normal mixture (SFNM) distribution learning, based on the objective criterion and learning curves. We applied all the methods to the same example and used the GRE value between

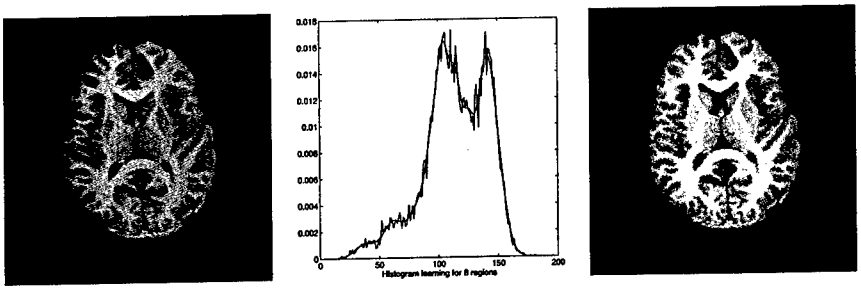


Figure 2: Results of MR brain tissue quantification and segmentation.

the image histogram and the estimated SFNM distribution as the goodness criterion to evaluate the quantification error. Figure 3 (a) shows learning curves of the PSOM and competitive learning (CL), averaged over 5 independent runs. As observed in the figure, PSOM outperforms CL learning by faster convergence and lower quantification error, and reaches a final GRE value of about 0.04 nats. Figure 3 (b) presents the comparison of PSOM with that of the EM algorithm for 25 epochs. As seen in the learning curves, PSOM algorithm again shows superior estimation performance. The final quantification error is about 0.02 nats while preserving the faster convergence rate.

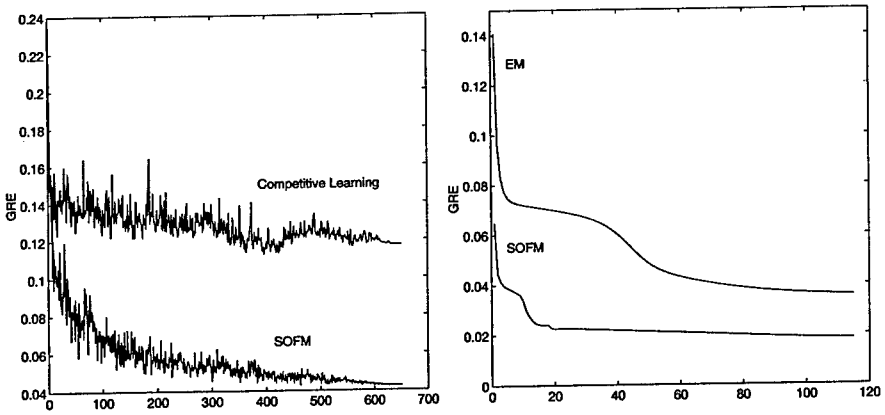


Figure 3: Comparison of the learning curves of PSOM and CL (left) and EM (right).

5. CONCLUSIONS

Our main contribution is the complete proposal of a three-step learning strategy for determination of both modular structure and components of the network. In this approach, the network structure (in terms of suitability

of the statistical model) is justified in the first step. It is followed by soft segmentation of data such that each data point supports all local components simultaneously. The associated probabilistic labels are then realized in the third step by competitive learning of this induced hard classification task. The main limitations of current approach are that, 1) it requires the testing of all possible network structure candidates during the model fitting procedure, hence is not efficient especially for processing MR sequence images where an on-line learning is preferred, and 2) applications to real MR data indicates the possibility of being trapped in a local minimum in ML estimation by the PSOM since there is no guarantee of attaining the global minimum. To summarize, the results of the experiments we have performed, indicate the plausibility of this approach for brain tissue analysis from MRI scans, and show that it can be applied to clinical problems such as those encountered in tissue segmentation and quantitative diagnosis.

References

[1] P. Santago and H. D. Gage, "Quantification of MR brain images by mixture density and partial volume modeling," *IEEE Trans. Med. Imag.*, Vol. 12, No. 3, pp. 566-574, September 1993.

[2] A. J. Worth and D. N. Kennedy, "Segmentation of magnetic resonance brain images using analog constraint satisfaction neural networks," *Information Processing in Medical Imaging*, pp. 225-243, 1993.

[3] K. S. Cheng, J. S. Lin, and C. W. Mao, "The application of competitive Hopfield neural network to medical image segmentation," *IEEE Trans. Med. Imaging*, Vol. 15, No. 4, pp. 560-567, August 1996.

[4] T. Adalı, *et al*, "Conditional distribution learning with neural networks and its application to channel equalization," to appear *IEEE Trans. Signal Processing.*

[5] L. Perlovsky and M. McManus, "Maximum likelihood neural networks for sensor fusion and adaptive classification," *Neural Networks*, Vol. 4, pp. 89-102, 1991.

[6] W. C. Lin, E. C. K. Tsao, and C. T. Chen, "Constraint satisfaction neural networks for image segmentation," *Pattern Recog.*, Vol. 25, pp. 679-693, 1992.

[7] M. Morrison and Y. Attikiouzel, "A probabilistic neural network based image segmentation network for magnetic resonance images," *Proc. Conf. Neural Nets.*, vol. 3, pp. 60-65, Baltimore, 1992.

[8] A. P. Dhawan and L. Arata, "Segmentation of medical images through competitive learning," *Comput. Meth. Prog. Biomed.*, Vol. 40, pp. 203-215, 1993.

[9] L. O. Hall, *et al*, "A comparison of neural network and fuzzy clustering techniques in segmenting magnetic resonance images of the brain," *IEEE Trans. Neural Nets.*, Vol. 3, pp. 672-682, 1992.

[10] S. Haykin, *Neural Networks: A Comprehensive Foundation.* New York: Macmillan College Publishing Company, 1994.

Author Index