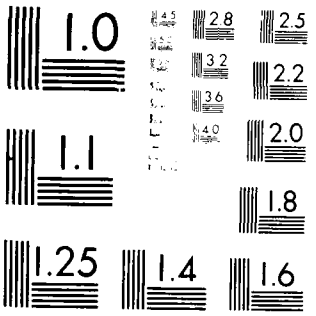
AD-A200 445 CONNECTIONIST MODELS FOR INTELLIGENT COMPUTATION(U) 1/1
MARYLAND UNIV COLLEGE PARK H H CHEN ET AL. 31 AUG 88
AFOSR-TR-88-1157 AFOSR-87-0388

UNCLASSIFIED F/G 12/9 NL

END
DATE FILMED
11-88



SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

AD-A200 445

DTIC ELECTE SEP 13 1988 S D CY

1b RESTRICTIVE MARKINGS: NONE	
2b DECLASSIFICATION / DOWNGRADING SCHEDULE: N/A	3 DISTRIBUTION / AVAILABILITY OF REPORT: Approved for public release; distribution unlimited.
4 PERFORMING ORGANIZATION REPORT NUMBER(S): 0388-1	5 MONITORING ORGANIZATION REPORT NUMBER(S): AFOSR-TR- 88-1157
6a NAME OF PERFORMING ORGANIZATION: Univ. of Maryland; 6b OFFICE SYMBOL (If applicable):	7a NAME OF MONITORING ORGANIZATION: AFOSR
6c ADDRESS (City, State, and ZIP Code): College Park, MD 20742	7b ADDRESS (City, State, and ZIP Code): Bldg 410 Bolling AFB, Washington, DC 20332-5260
8a NAME OF FUNDING / SPONSORING ORGANIZATION: AFOSR; 8b OFFICE SYMBOL (If applicable): NE	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: AFOSR-87-0388
8c ADDRESS (City, State, and ZIP Code): Bldg 410 Bolling AFB, Washington, DC 20332-5260	10 SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO 61102F; PROJECT NO 2305; TASK NO B1; WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)
Connectionist Models for Intelligent Computation

12 PERSONAL AUTHOR(S) H.H. Chen and Y.C. Lee

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Annual Technical	FROM 9/1/87 TO 8/31/88	8/31/88	

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

In the past year, we have 1) continued our investigation in the high-order neural network architectures 2) constructed a stereopsis network that is learned analytically and 3) proposed a novel scheme (PSIN) to automatically build up the network while learning.

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT: ☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION: UNCLASSIFIED
22a NAME OF RESPONSIBLE INDIVIDUAL	22b TELEPHONE (Include Area Code): (202) 767-4931; 22c OFFICE SYMBOL: NE

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted. All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE

Unclassified

AFOSR-TR- 88-1117

Project Title: Connectionist Models for Intelligent Computation

Contract/Grant No. AFOSR-87-0388

Contract/Grant Period of Performance: Sept. 1, 1987 - Aug. 31, 1988.

Principal Investigator	H.H. Chen	Tel #:	454-3182
	Y.C. Lee		454-3177

Research Objective: To study the underlying principles, architectures and applications of artificial neural networks for intelligent computations.

Approach: We use both numerical simulation and theoretical analysis to investigate various alternatives in connection shcemes, organization principles and architectures of artificial neural networks.

Progress for period 9/1/87 - 8/31/88:

1. We have continued our study of higher order neural networks. The superior processing power capacity and speed of the higher order neural network has been demonstrated for many tasks including text to speech, character recognition, noise removal, time series prediction etc. Currently, we are applying it to the speech recognition problem.

2. We have constructed a neural network to learn the task of stereopsis from random dot stereogram. The connection weights of the network are computed analytically from the Hebbion learning rule. The results show that the continuity and uniqueness constraints first proposed by Marr and Poggio are learned automatically.

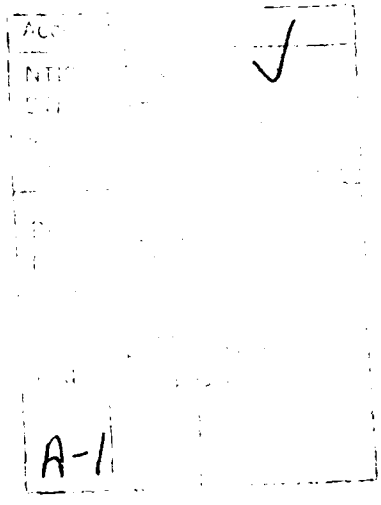
3. We proposed a novel scheme (PSIN) to automatically build a neural network while learning. The new scheme takes advantage of both the parallel and sequential strategies to solve a pattern classification or decision problem. We optimize an entropy measure to encourage the network to extract the best feature first to classify the pattern. Preliminary test of this new scheme shows that PSIN performs superior than the back propagation scheme in hard problems.

Publications

1. A Novel Net That Learns Sequential Decision Process, G.Z. Sun, Y.C. Lee, and H.H. Chen in "Neural Information Processing Systems" p. 760 (1988) Editor D. Anderson.

2. Learning Stereopsis with Neural Networks by G.Z. Sun, H.H. Chen, and Y.C. Lee, Proceeding IEEE First International Conference on Neural Networks, San Diego, California (1987), Vol IV, p. 345.

3. Parallel Sequential Induction Networks - New Paradigm of Neural Network Architecture by G,Z, Sun, H.H. Chen and Y.C. Lee, Proceeding IEEE Second Internation Conference on Neural Networks, San Diego (1988) Vol 1, 489.



LEARNING STEREOPSIS WITH NEURAL NETWORKS

G. Z. Sun, H. H. Chen, and Y. C. Lee

Laboratory for Plasma and Fusion Energy Studies and
Department of Physics and Astronomy and
Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742

ABSTRACT

A high order recursive neural network is constructed to learn the task of stereopsis from random dot stereograms. The connection weights of the network are learned through Hebbian rule. To avoid the problem of overwhelmingly large number of weights, we exploit the translational symmetry and trained only a small local patch and later transported uniformly to the whole network. Since the Hebbian learning is linear, the weights can be calculated analytically. The results show that the continuity and the uniqueness constraints first proposed by Marr and Poggio are learned automatically.

INTRODUCTION

Neural network models have been demonstrated to be very effective in computing perceptive problems such as vision, speech, and motor control.[1] One of its main advantages is the ability to learn automatically to perform a specific task by way of a learning algorithm.[2] In a single-layered perception, the error correction learning rule is guaranteed of convergence if a solution exists. In a multi-layered feed-forward network, back propagation of error messages is used to train the networks. These learning methods are generally nonlinear, making it hard to analyze the final network to understand the working principle of the networks.

In this paper, we would like to study an example, namely, the stereopsis of random dot images[3] wherein a recursive network is used. We would like to demonstrate that the algorithm for solving random dot stereogram can be learned by the simple Hebbian rule. Since the Hebbian rule involves neither error correction nor back propagation and is local and linear, it allows us to calculate the connection weights analytically. The result confirms the uniqueness and the continuity constraints that Marr and Poggio[4] first postulated to be in the working principle of the stereopsis network. Through Hebbian learning, these two constraints are learned automatically. Furthermore, the weights learned are symmetrical which ensures the convergence of the recursive network. The network we used has five depth layers. Each layer contains $100 \times 100 = 10^4$ pixels of neuron cells. If fully connected, it would require 2.5×10^9 connection

weights, a formidably large number. To reduce from this large number of weights, we exploit the localness and the translational symmetry of the stereopsis problem. We train a small patch of the network with pixel size $a \times a$ ($a \lesssim 11$) and then transport it to the whole image plane. The result is a high order recursive network with local connection weights uniformly distributed in the network.

II. TRAINING OF THE NETWORK

The stereo vision is achieved by detecting the binocular disparity of the two images observed by the two eyes. The random dot stereogram demonstrated that the stereo perception is an early vision problem that does not involve the high level task of recognition and identification of visual objects. In Fig. 1(a) and (b) we show two random dot images of the left and the right eyes. Each has 100×100 pixels with an equal probability ($\nu = 1/2$) to be white and black. The actual three-dimensional image, a five-layered cake, is shown in Fig. 1(c). Our task is to train a neural network to construct the three-dimensional image surfaces from the two monocular random dot images. Marr and Poggio constructed a network with connection weights designed from the two constraints, namely the uniqueness constraints which says that in any given direction we see only one image surface and the continuous constraint that says a surface is usually continuous. In this paper we are going to demonstrate that these two working principles of stereopsis can be learned automatically by the networks using Hebbian learning rule.

In order to proceed, we first choose the following conventions for notations and representations.

(i) In the random dot images, a black dot is assigned the value +1 and a white dot −1.

(ii) In each monocular image the probability for a dot to be black is ν.

(iii) The maximum depth is an integer D.

(iv) The input to the network is a conjunction of the left and the right monocular images (see Fig. 1(d)),

$$I_{i,\underset{\sim}{k}} = (RL)_{i,k_x,k_y} \equiv R_{k_x,k_y} \cdot L_{k_x-i+1,k_y} , \qquad (1)$$

where $\underset{\sim}{k} = (k_x,k_y)$ is the position vector of the pixel point on a single monocular image, and $i = 1,2,\ldots D$ is the index for the different depth level countered from the bottom (i=1) to the top (i=D). R and L denote the right and the left monocular images respectively. Equation (1) implies that the input is composed of all possible matches between the left and the right monocular images including both the black and the white dots.

(v) The output of the network also consists of five (D=5) layers of planes with size 100 × 100. We let

$$S_{i,\underset{\sim}{k}} = \begin{cases} s & (i,\underset{\sim}{k}) \text{ lies on a solution surface} \\ -1 & \text{otherwise} \end{cases} \text{ if } .$$

Here the solution surface means the outline surface of the 3-D object viewed from above (as shown in Fig. 1(c)) and s is a normalization factor such that the total sum of all the components of the connection weight would be zero if s = D − 1.

We then use the Hebbian learning rule to construct the weights between $S_{i,\underset{\sim}{k}}$ and $I_{j,\underset{\sim}{k}'}$,

$$W_{i,\underset{\sim}{k};j,\underset{\sim}{k}'} = \sum_r S^r_{i,\underset{\sim}{k}} I^r_{j,\underset{\sim}{k}'} \equiv Np\langle S_{i,\underset{\sim}{k}} I_{j,\underset{\sim}{k}'}\rangle , \tag{2}$$

where the summation is over all possible pattern pairs of S and I, and N_p is the total number of patterns.

For training patterns, we choose a small patch of size a×a with a << 100 to reduce the number of weights to a manageable size. The training is done by dividing the patch into four frontal planes with two dividing lines, one horizontal and another vertical. The position of these two lines and the height of the four sub-patches are uniformly random.

To calculate the analytical weights, we note the following:

(i) $$S_{i,\underset{\sim}{k}} = \begin{cases} s & (i,\underset{\sim}{k}) \text{ lies on the solution plane} \\ 1 & \text{otherwise} \end{cases} \text{ if } , \tag{3}$$

(ii) $$(RL)_{i,\underset{\sim}{k}} = \begin{cases} 1 & \text{if } (i,\underset{\sim}{k}) \text{ lies on the solution plane} \\ 1 & \text{with prob. } p(1) = \nu + (1-\nu) \\ -1 & \text{with prob. } p(-1) = 2\nu(1-\nu) \end{cases} \Big\} \text{ if not on solution plane,} \tag{4}$$

(iii) if $(i,\underset{\sim}{k})$ is not on the solution plane, we have the ensemble average

$$\langle (RL)_{i,\underset{\sim}{k}}\rangle = (1-2\nu)^2 , \tag{5}$$

(iv) with the patch a×a, two points (k_x,k_y) and $(k_x+x\ ,\ k_y+y)$ would lie on the same sub-plane with the probability

$$P_d = \frac{1}{(a-1)^2}(a-1-|x|)(a-1-|y|) , \tag{6}$$

where we have assumed that (k_x,k_y) is at the center of the patch (an approximation justified by the translational symmetry). To proceed, we

consider two exclusive cases $i = j$ and $i \neq j$.

(a) diagonal $i = j$,

$$W_{i,\underset{\sim}{k};i,\underset{\sim}{k}+\underset{\sim}{r}} \equiv \langle S_{i,\underset{\sim}{k}}(RL)_{i,\underset{\sim}{k}+\underset{\sim}{r}} \rangle \ . \tag{7}$$

Since $S_{i,\underset{\sim}{k}}$ can assume the value s with probability $1/D$ and the value -1 with probability $(D-1)/D$, we have

$$\begin{aligned} \langle S_{i,\underset{\sim}{k}}(RL)_{i,\underset{\sim}{k}+\underset{\sim}{r}} \rangle &= \frac{s}{D} \langle (RL)_{i,\underset{\sim}{k}+\underset{\sim}{r}} \rangle S_{i,\underset{\sim}{k}} = s \\ &\quad - \frac{D-1}{D} \langle (RL)_{i,\underset{\sim}{k}+\underset{\sim}{r}} \rangle S_{i,\underset{\sim}{k}}=-i \ , \end{aligned} \tag{8}$$

when $(i,\underset{\sim}{k})$ lies on a solution plane, the probability that $(i,\underset{\sim}{k}+\underset{\sim}{r})$ is not on any solution plane would be

$$(1-P_d)(1-1/D) \ ,$$

and we have

$$\begin{aligned} &\langle (RL)_{i,\underset{\sim}{k}+\underset{\sim}{r}} \rangle S_{i,\underset{\sim}{k}}=s \\ &= (1-P_d)(1-1/D)(1-2\nu)^2 + 1 - (1-P_d)(1-1/D) \ . \end{aligned} \tag{9}$$

Similarly, we have

$$\langle (RL)_{i,\underset{\sim}{k}+\underset{\sim}{r}} \rangle S_{i,\underset{\sim}{k}}=-1 = (1-P_d)\frac{1}{D} + \left[1 - \frac{1}{D}(1-P_d)\right](1-2\nu)^2 \ . \tag{10}$$

Finally, we have

$$W_{i,\underset{\sim}{k};i,\underset{\sim}{k}+\underset{\sim}{r}} = N_p\left\{\frac{s+1}{D^2} 4\nu(1-\nu)(D-1)P_d + \frac{s-D+1}{D}\left[1-4\nu(1-\nu)\left(1-\frac{1}{D}\right)\right]\right\} . \tag{11}$$

(b) $i \neq j$. In this case $(i,\underset{\sim}{k})$ and $(j,\underset{\sim}{k}+\underset{\sim}{r})$ can not lie on the same plane. If $(i,\underset{\sim}{k})$ lies on a solution plane, the probability for $(j,\underset{\sim}{k}+\underset{\sim}{r})$ to be also on a solution plane is

$$P_1 = \frac{1}{D}(1 - P_d) \ . \tag{12}$$

On the other hand, if $(i,\underset{\sim}{k})$ is not on a solution plane, then the probability that $(j,\underset{\sim}{k}+\underset{\sim}{r})$ would lie on a solution plane is

$$P_2 = P_d \frac{1}{D-1} + (1-P_d)\frac{1}{D} = \frac{1}{D}\left[1+P_d/(D-1)\right] \ . \tag{13}$$

With these probabilities, we can calculate

$$\langle (RL)_{j,\underset{\sim}{k}+\underset{\sim}{r}} \rangle_{S_{i,\underset{\sim}{k}}=s} = P_1 + (1-P_1)(1-2\nu)^2 \tag{14}$$

and

$$\langle (RL)_{j,\underset{\sim}{k}+\underset{\sim}{r}} \rangle_{S_{i,\underset{\sim}{k}}=-1} = P_2 + (1-P_2)(1-2\nu)^2 \ . \tag{15}$$

Therefore when $i \neq j$, we have

$$W_{i,\underset{\sim}{k};\ j,\underset{\sim}{k}+\underset{\sim}{r}} = N_p\{-\frac{s+1}{D^2} 4\nu(1-\nu)P_d + \frac{s-D+1}{D}[1-4\nu(1-\nu)(1-\frac{1}{D})]\} \ . \tag{16}$$

Combining Eqs. (11) and (16), we have for general i and j,

$$\begin{aligned} W_{i,\underset{\sim}{k},j,\underset{\sim}{k}+\underset{\sim}{r}} &= N_p \frac{s+1}{D^2} 4\nu(1-\nu)(D\delta_{ij}-1)P_d \\ &+ N_p \frac{s-D+1}{D}[1-4\nu(1-\nu)(1-\frac{1}{D})] \ . \end{aligned} \tag{17}$$

The second term in Eq. (17) can be ignored if we choose $s = D - 1$. Substituting the expression of P_d from Eq. (6), we get

$$W_{i,\underset{\sim}{k};j,\underset{\sim}{k}'} = \frac{N_p}{D} \frac{4\nu(1-\nu)}{(a-1)^2} (a-1-|k_x-k_x'|)(a-1-|k_y-k_y'|)(D\delta_{ij}-1) \ , \tag{18}$$

where k_x, k_y is chosen as the center of a patch axa and k_x', k_y' run through the whole patch. It is easily seen that this weight matrix is symmetrical between i and j. This is a consequence of the translation and reflection symmetry in our training patterns. The last factor in Eq. (18) also shows that the weights between neurons on the same layer are excitatory while for neurons on different layers are inhibitory. A direct confirmation of the continuity and the uniqueness constraint is proposed by Marr and Poggio.

III. CONVERGENCE THEOREM

Hopfield is the first to show the convergence of an asynchronous symmetrical recursive network with the help of a Liapunov function. In our problem, the weights are symmetrical and we can implement the dynamics of our network with two schemes, a) the maximum scheme and b) the threshold scheme. Their performance is similar. As a matter of fact, in order to enhance the stability of our result, we choose to add a forcing from the input to our network which can be integrated into the convergence theorem easily.

The maximum scheme evolves according to

$$X_{i,\underset{\sim}{k}}^{t+1} = \max_{1\le i\le D}\left[\sum_{j,\underset{\sim}{k}'} W_{i,\underset{\sim}{k};j,\underset{\sim}{k}'}\, X_{j,\underset{\sim}{k}'}^{t} + \sigma X_{i,\underset{\sim}{k}}^{0}\right], \tag{19}$$

where

$$\max_{1\le i\le D}(y_i) = \begin{cases} 1 & \text{if } y_i = \max(y_j;\ j=1,\ldots D) \\ -1 & \text{otherwise} \end{cases},$$

and $X_{i,k}^{0}$ is the input state of the neurons. Following Goles and Vichniac,[5] we can show that this scheme is convergent with the help of the Liapunov function,

$$E_M(X^t, X^{t+1}) = \sum_{i,\underset{\sim}{k}} \sum_{j,\underset{\sim}{k}'} X_{i,\underset{\sim}{k}}^{t+1} W_{i,\underset{\sim}{k};j,\underset{\sim}{k}'} X_{j,\underset{\sim}{k}'}^{t} + \sum_{i,\underset{\sim}{k}} (X_{i,\underset{\sim}{k}}^{t+1} + X_{i,\underset{\sim}{k}}^{t})\sigma X_{i,k}^{0}, \tag{20}$$

where X^t and X^{t+1} are the neuron states at time step t and t+1, respectively. It is then straightforward to show that the updating monotonically increases the Liapunov function (20) and therefore guarantees its convergence.

The advantage of the maximum scheme is its simplicity. It ensures that the uniqueness constraint is satisfied strictly. However, since we are interested in the learning of the network to automatically implement the uniqueness constraint, we are also interested in the threshold scheme which takes the mutual inhibition of the pixels along line of sight into account. The threshold scheme evolves according to

$$X_{i,\underset{\sim}{k}}^{t+1} = \theta\left[\sum_{j,\underset{\sim}{k}'} W_{i,\underset{\sim}{k};j\cdot\underset{\sim}{k}'} X_{j,\underset{\sim}{k}'}^{t} + \sigma X_{i,\underset{\sim}{k}}^{0} + \tau\right], \tag{21}$$

where θ is the step function defined as

$$\theta(x) = \begin{cases} 1 & x > 0 \\ -1 & \text{if } x \le 0 \end{cases}$$

and τ is a threshold value. This scheme is also ensured of convergence. For synchronous updating, the Liapunov function is very similar to Eq. (20) and is given by[5]

$$E_T(X^t, X^{t+1}) = \sum_{i,\underset{\sim}{k}} X_{i,\underset{\sim}{k}}^{t+1} W_{i,\underset{\sim}{k};j\underset{\sim}{k}'} X_{j,\underset{\sim}{k}'}^{t} + \sum_{i,\underset{\sim}{k}} (X_{i,\underset{\sim}{k}}^{t+1} + X_{i,\underset{\sim}{k}}^{t})(\sigma X_{i,\underset{\sim}{k}}^{0} + \tau). \tag{22}$$

Again, we can show that $\Delta E_T \geq 0$ strictly. Both of the maximum and the threshold scheme have been implemented and show similar performance.

IV. STATIONARY STATES AND THE NUMERICAL IMPLEMENTATION

After proving the convergence theorem, the next question to ask is where would the network converge to? Unfortunately, the complete discussion of the attractor states are very tedious and is not the main interest of this paper. We would only state that the considerations of the following questions help us to find the optimal choice of the few parameters remain in our network, namely the threshold value, the strength of the forcing by initial pattern, and the size of our training patch a.

The questions involved are:

(i) Are the internal cells stationary? An internal cell is a pixel cell whose immediate neighbors are occupied by cells with the same value.

(ii) Are the cells at a boundary stationary? As one solution surface meets another solution surfacae at a boundary, this boundary must be stationary.

(iii) The corner must be stationary. This is a tougher condition to be met than (ii).

(iv) Isolated dots are false and should be eliminated.
These considerations lead us to the choice of $3 \leq a \lesssim 13$.

We have run many numerical simulations of our networks with the above a values. Typically, three or four iterations are sufficient to attain the final stationary state (see Fig. 2). Sometimes, merely one or two iterations already give an almost perfect result. The remaining iterations serve only to the removing of a few isolated false spots. Among the results obtained in terms of different size and weights, the one obtained with a = 5 has the cleanest shape and sharpest corners.

We have also tested our network with 3-D images other than rectangles, for instance, triangles and octagons. The learned weights generalize well for these cases (see Fig. 3). We also tested the robustness of the network by implementing the numerically acquired weights. The performance is as good as those with the analytical weights.

V. CONCLUSION AND DISCUSSIONS

We demonstrated in this paper that using high order recursive network the connection weights can be learned automatically from Hebb's rule to perform stereopsis on random dot stereograms. The Hebb's rule is linear that allows us to construct the weights analytically. The results showed that the continuity and the uniqueness consraint first proposed by Marr and Poggio are learned automatically. The weights obtained are symmetrical and

therefore guarantee the convergence of the network. Numerical implementations confirmed these predictions.

REFERENCES

1. See, for example, AIP Proceedings 151, Neural Networks for Computing, Snowbird, Utah (1986), ed. John Denker.
2. D. Rummelhart and J. McClelland, "Parallel Distributive Processing", MIT Press (1986).
3. D. Marr, "Vision", Freeman (1982).
4. D. Marr and T. Poggio, Science 194, 283 (1976).
5. E. Goles and G. Y. Vichniac, "Liapunov function for parallel neuron network," Proceedings of Conference on the Neuron Network Computing, Snowbird, 1986.

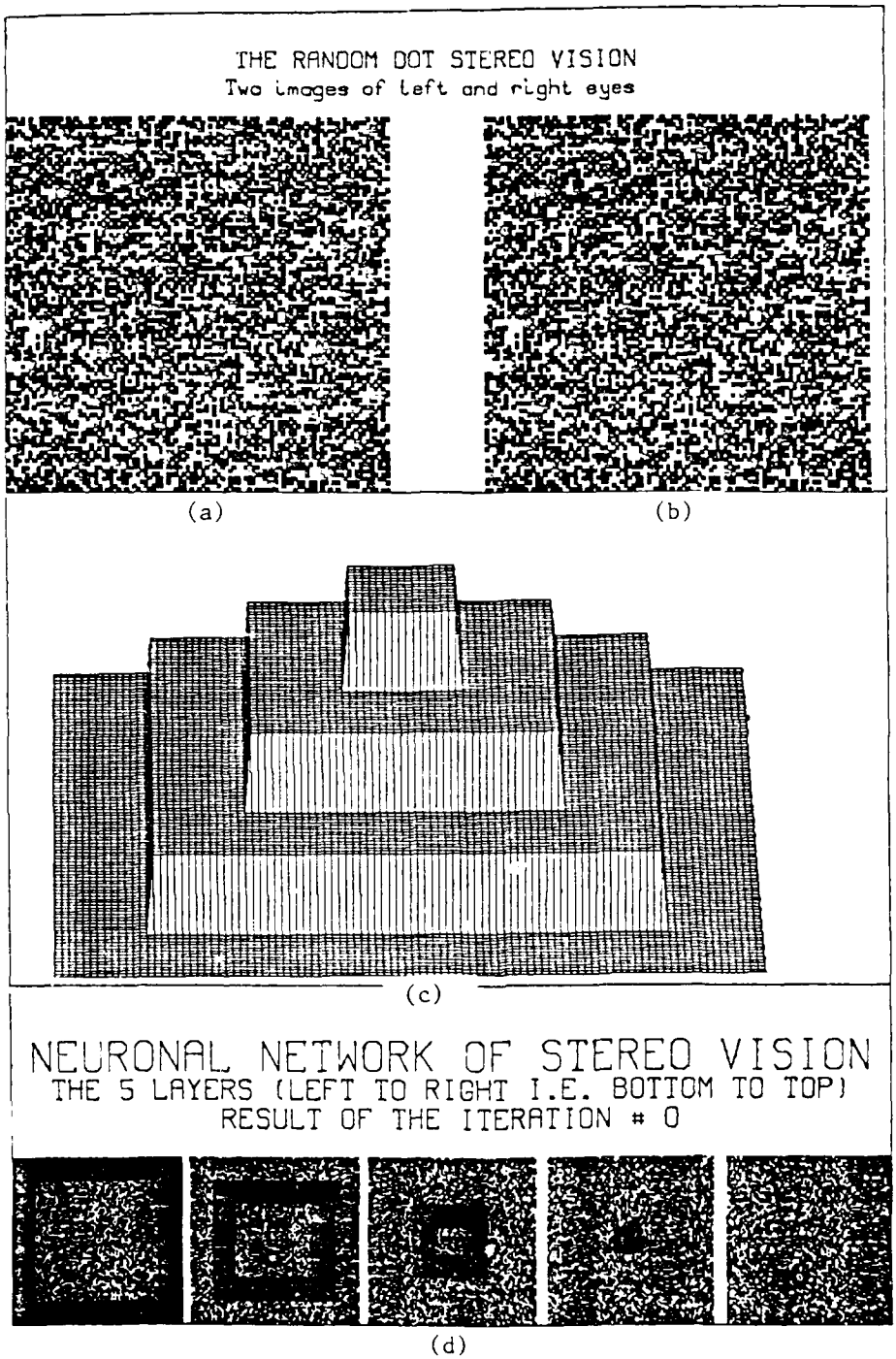

THE RANDOM DOT STEREO VISION
Two images of left and right eyes
(a)
(b)
(c)
NEURONAL NETWORK OF STEREO VISION
THE 5 LAYERS (LEFT TO RIGHT I.E. BOTTOM TO TOP)
RESULT OF THE ITERATION # 0
(d)

Figure 1

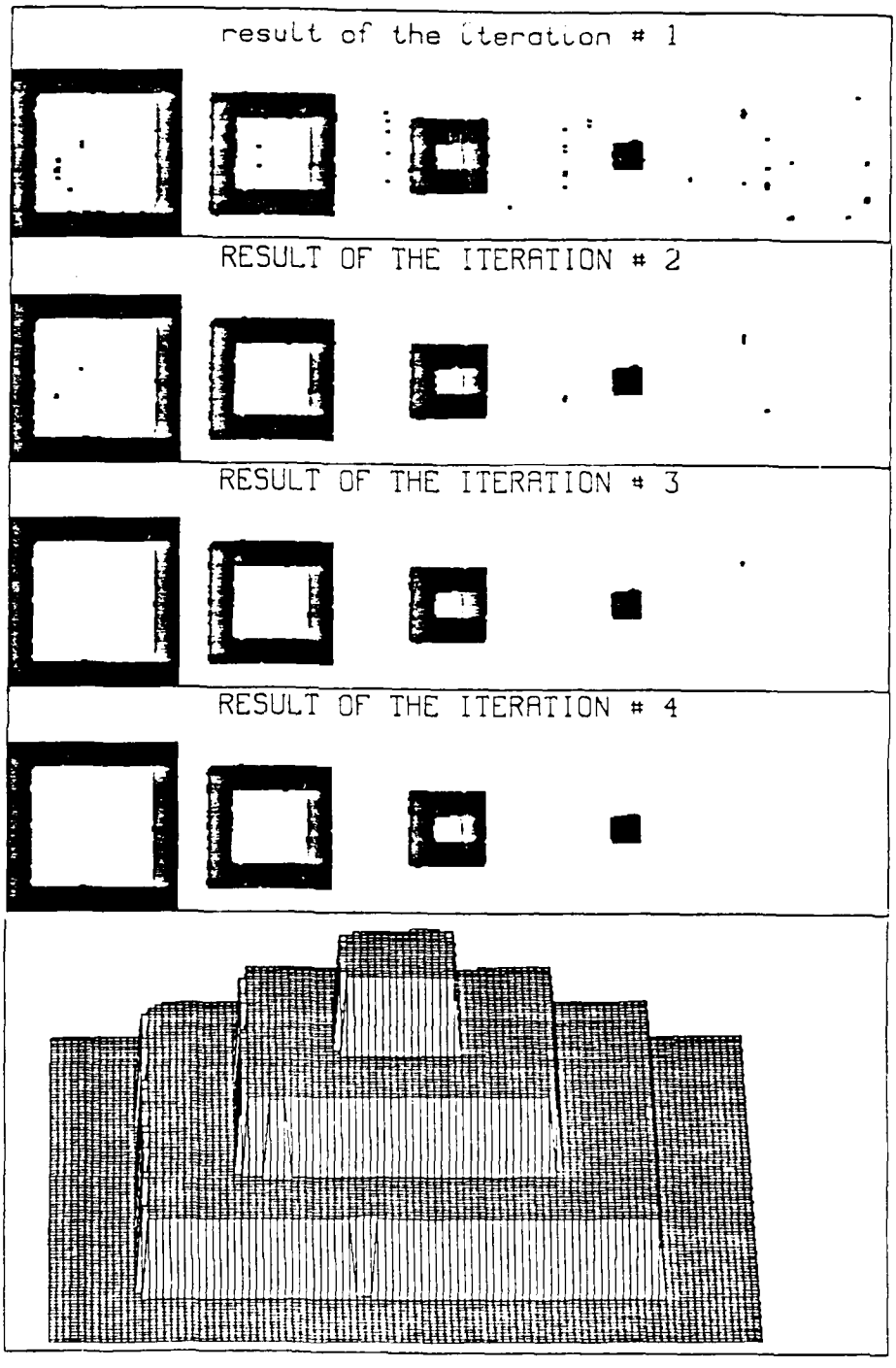

result of the iteration # 1
RESULT OF THE ITERATION # 2
RESULT OF THE ITERATION # 3
RESULT OF THE ITERATION # 4

Figure 2

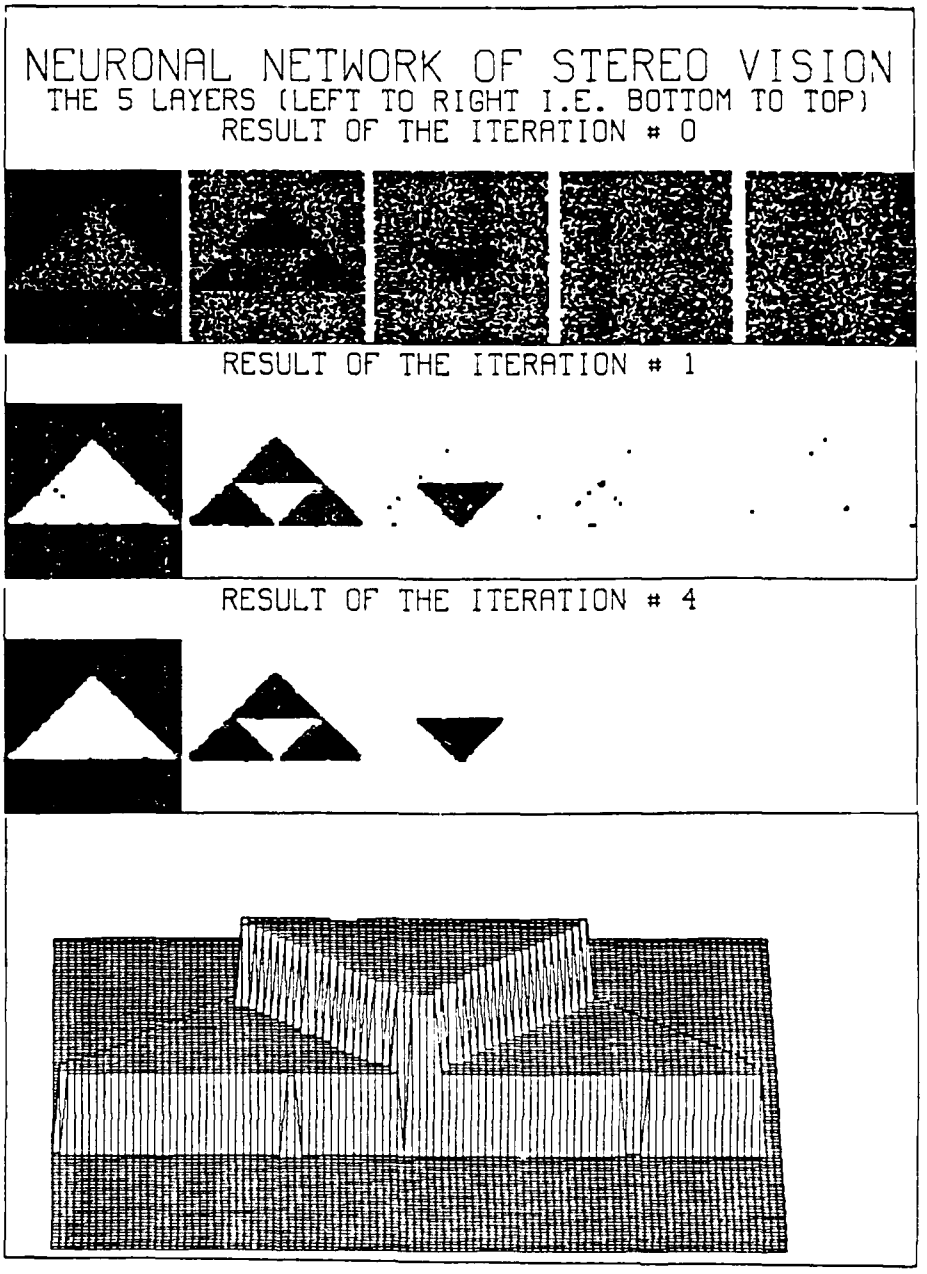

NEURONAL NETWORK OF STEREO VISION
THE 5 LAYERS (LEFT TO RIGHT I.E. BOTTOM TO TOP)
RESULT OF THE ITERATION # 0
RESULT OF THE ITERATION # 1
RESULT OF THE ITERATION # 4

Figure 3

A NOVEL NET THAT LEARNS SEQUENTIAL DECISION PROCESS

G.Z. SUN, Y.C. LEE and H.H. CHEN

Department of Physics and Astronomy
and
Institute for Advanced Computer Studies
UNIVERSITY OF MARYLAND,COLLEGE PARK,MD 20742

ABSTRACT

We propose a new scheme to construct neural networks to classify patterns. The new scheme has several novel features :

1. We focus attention on the important attributes of patterns in ranking order. Extract the most important ones first and the less important ones later.
2. In training we use the information as a measure instead of the error function.
3. A multi-perceptron-like architecture is formed auomatically. Decision is made according to the tree structure of learned attributes.

This new scheme is expected to self-organize and perform well in large scale problems.

1 INTRODUCTION

It is well known that two-layered perceptron with binary connections but no hidden units is unsuitable as a classifier due to its limited power [1]. It cannot solve even the simple *exclusive-or* problem. Two extensions have been proposed to remedy this problem. The first is to use higher order connections [2]. It has been demonstrated that high order connections could in many cases solve the problem with speed and high accuracy [3], [4]. The representations in general are more local than distributive. The main drawback is however the combinatorial explosion of the number of high-order terms. Some kind of heuristic judgement has to be made in the choice of these terms to be represented in the network.

A second proposal is the multi-layered binary network with hidden units [5]. These hidden units function as features extracted from the bottom input layer to facilitate the classification of patterns by the output units. In order to train the weights, learning algorithms have been proposed that back-propagate the errors from the visible output layer to the hidden layers for eventual adaptation to the desired values. The multi-layered networks enjoy great popularity in their flexibility.

However, there are also problems in implementing the multi-layered nets. Firstly, there is the problem of allocating the resources. Namely, how many hidden units would be optimal for a particular problem. If we allocate too many, it is not only wasteful but also could negatively affect the performance of the network. Since too many hidden units implies too many free parameters to fit specifically the training patterns. Their ability to generalize to noval test patterns would be adversely affected. On the other hand, if too few hidden units were allocated then the network would not have the power even to represent the trainig set. How could one judge beforehand how many are needed in solving a problem? This is similar to the problem encountered in the high order net in its choice of high order terms to be represented.

Secondly, there is also the problem of scaling up the network. Since the network represents a parallel or coorperative process of the whole system, each added unit would interact with every other units. This would become a serious problem when the size of our patterns becomes large.

Thirdly, there is no sequential communication among the patterns in the conventional network. To accomplish a cognitive function we would need the patterns to interact and communicate with each other as the human reasoning does. It is difficult to envision such an interacton in current systems which are basically input-output mappings.

2 THE NEW SCHEME

In this paper, we would like to propose a scheme that constructs a network taking advantages of both the parallel and the sequential processes.

We note that in order to classify patterns, one has to extract the intrinsic features, which we call attributes. For a complex pattern set, there may be a large number of attributes. But differnt attributes may have different

ranking of importance. Instead of extracing them all simultaneously it may be wiser to extract them sequentially in order of its importance [6], [7]. Here the importance of an attribute is determined by its ability to partition the pattern set into sub-categories. A measure of this ability of a processing unit should be based on the extracted information. For simplicity, let us assume that there are only two categories so that the units have only binary output values 1 and 0 (but the input patterns may have analog representations). We call these units, including their connection weights to the input layer, *nodes*. For given connection weights, the patterns that are classified by a *node* as in category 1 may have their true classifications either 1 or 0. Similarly, the patterns that are classified by a *node* as in category 0 may also have their true classifications either 1 or 0. As a result, four groups of patterns are formed: (1,1), (0,0), (1,0), (0,1). We then need to judge on the efficiency of the *node* by its ability to split these patterns optimally. To do this we shall construct the impurity fuctions for the *node*. Before splitting, the impurity of the input patterns reaching the node is given by

$$I_b = -P_1^b \log P_1^b - P_0^b \log P_0^b \tag{1}$$

where $P_1^b = N_1^b/N$ is the probability of being truely classified as in category 1, and $P_0^b = N_0^b/N$ is the probability of being truely classified as in category 0. After splitting, the patterns are channelled into two branches, the impurity becomes

$$I_a = -P_1^a \sum_{j=0,1} P(j,1) \log P(j,1) - P_0^a \sum_{j=0,1} P(j,0) \log P(j,0) \tag{2}$$

where $P_1^a = N_1^a/N$ is the probability of being classified by the node as in category 1, $P_0^a = N_0^b/N$ is the probability of being classified by the node as in category 0, and $P(j,i)$ is the probability of a pattern, which should be in category j, but is classified by the node as in category i. The difference

$$\Delta I = I_b - I_a \tag{3}$$

represents the decrease of the impurity at the node after splitting. It is the quantity that we seek to optimize at each node. The logarithm in the impurity function come from the information entropy of Shannon and Weaver. For all practical purpose, we found the optimization of (3) the same as maximizing the entropy [6]

$$S = \frac{N_1}{N}\left[(\frac{N_{01}}{N_1})^2 + (\frac{N_{11}}{N_1})^2\right] + \frac{N_0}{N}\left[(\frac{N_{00}}{N_0})^2 + (\frac{N_{10}}{N_0})^2\right] \tag{4}$$

where N_i is the number of training patterns classified by the node as in category i, N_{ij} is the number of training patterns with true classification in category i but classified by the node as in category j. Later we shall call the terms in the first bracket S_1 and the second S_2. Obviously, we have

$$N_i = N_{0i} + N_{1i}, \qquad i = 0, 1$$

After we trained the first unit, the training patterns were split into two branches by the unit. If the classificaton in either one of these two branches is pure enough, or equivalently either one of S_1 and S_2 is fairly close to 1, then we would terminate that branch (or branches) as a leaf of the decision tree. and classify the patterns as such. On the other hand, if either branch is not pure enough, we add additional node to split the pattern set further. The subsequent unit is trained with only those patterns channeled through this branch. These operations are repeated until all the branches are terminated as leaves.

3 LEARNING ALGORITHM

We used the stochastic gradient descent method to learn the weights of each node. The training set for each node are those patterns being channeled to this node. As stated in the previous section, we seek to maximize the entropy function S. The learning of the weights is therefore conducted through

$$\Delta W_j = \eta \frac{\partial S}{\partial W_j} \tag{5}$$

Where η is the learning rate. The gradient of S can be calculated from the following equation

$$\frac{\partial S}{\partial W_j} = \frac{1}{N}\Big[(1-2\frac{N_{01}^2}{N_1^2})\frac{\partial N_{11}}{\partial W_j} + (1-2\frac{N_{11}^2}{N_1^2})\frac{\partial N_{01}}{\partial W_j} + (1-2\frac{N_{00}^2}{N_0^2})\frac{\partial N_{10}}{\partial W_j} + (1-2\frac{N_{10}^2}{N_0^2})\frac{\partial N_{00}}{\partial W_j}\Big] \tag{6}$$

Using analog units

$$O^r = \frac{1}{1+exp(-\sum_j W_j I_j^r)} \tag{7}$$

we have

$$\frac{\partial O^r}{\partial W_j} = O^r(1-O^r)I_j^r \tag{8}$$

Furthermore, let $A^r = 1$ or 0 being the true answer for the input pattern r , then

$$N_{ij} = \sum_{r=1}^{N}\Big[iA^r + (1-i)(1-A^r)\Big]\Big[jO^r + (1-j)(1-O^r)\Big] \tag{9}$$

Substituting these into equation (5), we get

$$\Delta W_j = 2\eta \sum_r \Big[2A^r(\frac{N_{11}}{N_1} - \frac{N_{10}}{N_0}) + \frac{N_{10}^2}{N_0^2} - \frac{N_{11}^2}{N_1^2}\Big]O^r(1-O^r)I_j^r \tag{10}$$

In applying the formula (10),instead of calculating the whole summation at once, we update the weights for each pattern individually. Meanwhile we update N_{ij} in accord with equation (9).

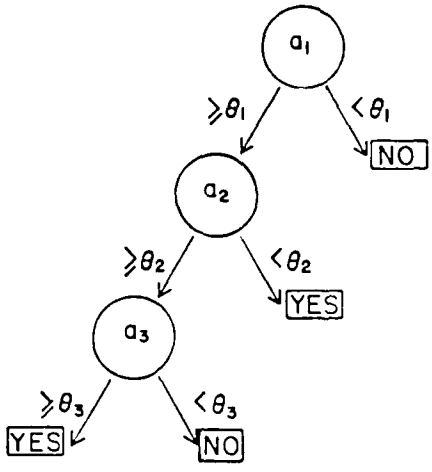


Figure 1: *The given classification tree, where θ_1, θ_2 and θ_3 are chosen to be all zeros in the numerical example.*

4 AN EXAMPLE

To illustrate our method, we construct an example which is itself a decision tree. Assuming there are three hidden variables a_1, a_2, a_3, a pattern is given by a ten-dimensional vector $I_1, I_2, \ldots, I_{10}$, constructed from the three hidden variables as follows

$$
\begin{aligned}
I_1 &= a_1 + a_3 & \qquad I_6 &= 2a_3 \\
I_2 &= 2a_1 - a_2 & \qquad I_7 &= a_3 - a_1 \\
I_3 &= a_3 - 2a_2 & \qquad I_8 &= 2a_1 + 3a_3 \\
I_4 &= a_1 + 2a_2 + 3a_3 & \qquad I_9 &= 4a_3 - 3a_1 \\
I_5 &= 5a_1 - 4a_4 & \qquad I_{10} &= 2a_1 + 2a_2 + 2a_3.
\end{aligned}
$$

A given pattern is classified as either 1 (yes) or 0 (no) according to the corresponding values of the hidden variables a_1, a_2, a_3. The actual decision is derived from the decision tree in *Fig.*1.

In order to learn this classification tree, we construct a training set of 5000 patterns generated by randomly chosen values a_1, a_2, a_3 in the interval -1 to +1. We randomly choose the initial weights for each node, and terminate

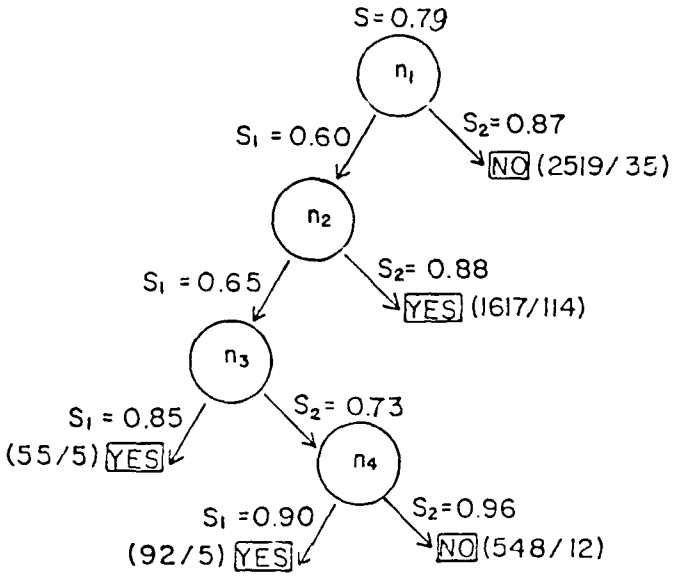


Figure 2: *The learned classification tree structure*

a branch as a leaf whenever the branch entropy is greater than 0.80. The entropy is started at $S = 0.65$, and terminated at its maximum value $S = 0.79$ for the first node. The two branches of this node have the entropy fuction valued at $S_1 = 0.61, S_2 = 0.87$ respectively. This corrosponds to 2446 patterns channeled to the first branch and 2554 to the second. Since $S_2 > 0.80$ we terminate the second branch. Among 2554 patterns channeled to the second branch there are 2519 patterns with true classification as *no* and 35 *yes* which are considered as errors. After completing the whole training process, there are totally four nodes automatically introduced. The final result is shown in a tree structure in *Fig.2*.

The total errors classified by the learned tree are 3.4 % of the 5000 trainig patterns. After trainig we have tested the result using 10000 novel patterns, the error among which is 3.2 %.

5 SUMMARY

We propose here a new scheme to construct neural network that can automatically learn the attributes sequentially to facilitate the classification of patterns according to the ranking importance of each attribute. This scheme uses information as a measure of the performance of each unit. It is

self-organized into a presumably *optimal* structure for a specific task. The sequential learning procedure focuses attention of the network to the most important attribute first and then branches out to the less important attributes. This strategy of searching for attributes would alleviate the scale up problem forced by the overall parallel back-propagation scheme. It also avoids the problem of resource allocation encountered in the high-order net and the multi-layered net. In the example we showed the performance of the new method is satisfactory. We expect much better performance in problems that demand large size of units.

6 acknowledgement

This work is partially supported by AFOSR under the grant 87-0388.

References

[1] M. Minsky and S. Papert, *Perceptron*, MIT Press Cambridge, Ma(1969).

[2] Y.C. Lee, G. Doolen, H.H. Chen, G.Z. Sun, T. Maxwell, H.Y. Lee and C.L. Giles, *Machine Learning Using A High Order Connection Netweork*, Physica **D22**,776-306 (1986).

[3] H.H. Chen, Y.C. Lee, G.Z. Sun, H.Y. Lee, T. Maxwell and C.L. Giles, *High Order Connection Model For Associate Memory*, AIP Proceedings Vol.151,p.86, Ed. John Denker (1986).

[4] T. Maxwell, C.L. Giles, Y.C. Lee and H.H. Chen, *Nonlinear Dynamics of Artificial Neural System*, AIP Proceedings Vol.151,p.299, Ed. John Denker(1986).

[5] D. Rummenlhart and J. McClelland, *Parallel Distributive Processing*, MIT Press(1986).

[6] L. Breiman, J. Friedman, R. Olshen, C.J. Stone, *Classification and Regression Trees*,Wadsworth Belmont, California(1984).

[7] J.R. Quinlan, *Machine Learning*, Vol.1 No.1(1986).

UMIACS–TR–88–26
CS–TR–2013

April, 1988

Parallel Sequential Induction Network a New Paradigm of Neural Network Architecture

G.Z. Sun
Laboratory for Plasma Physics Research
Department of Physics and Astronomy

H.H. Chen and Y.C. Lee

Institute for Advanced Computer Studies,
Laboratory for Plasma Physics Research and
Department of Physics and Astronomy
University of Maryland
College Park, MD 20742

ABSTRACT

We present here a new scheme to automatically construct a neural network architecture that takes advantage of both the parallel and sequential strategies to solve a pattern classification or decision problem. The new scheme optimizes an entropy measure to train nodes that extract attributes from the training patterns. The sequential extraction of attributes with ranking order could alleviate significantly the scale up problem of an all parallel network. Examples of decision tree problem demonstrate amply the superior performance of PSIN (Parallel Sequential Induction Network) against the usual back propagation procedure in multi–layered networks.

PARALLEL SEQUENTIAL INDUCTION NETWORK
A NEW PARADIGM OF NEURAL NETWORK ARCHITECTURE

G.Z. Sun, H.H. Chen and Y.C. Lee

Laboratory for Plasma Physics Research,
Department of Physics and Astronomy
and
Institute for Advanced Computer Studies
UNIVERSITY OF MARYLAND,COLLEGE PARK,MD 20742

ABSTRACT

We present here a new scheme to automatically construct a neural network architecture that takes advantage of both the parallel and sequential strategies to solve a pattern classification or decision problem. The new scheme optimizes an entropy measure to train nodes that extract attributes from the training patterns. The sequential extraction of attributes with ranking order could alleviate significantly the scale up problem of an all parallel network. Examples of decision tree problems demonstrate amply the superior performance of PSIN (Parallel Sequential Induction Network) against the usual back propagation procedure in multi-layered networks.

1 Introduction

When we make up the decision on a certain task, usually we have to evaluate a number of factors called attributes here. These attributes, in general, have different importance in helping us to make up the decision. As a matter of fact, some of the attributes are independent of each other, and can be evaluated at the same time, i.e. in parallel, while other attributes may have to wait until some preliminary decision based on more important attributes on a subtask has already been made. This combination of parallel and sequential strategies in the decision processes can

be very efficient indeed [1],[2]. On the other hand, we note that the neural network studied so far employs mostly the parallel strategy [3]. These are just input output mappings, with the neural network serving the role of a nonlinear mapping function. There are two major limitations of this approach. First, it is difficult to identify the proper network structure for a given task. Second, the scaling up problem and the associated low learning speed are serious drawbacks of these networks. These limitations could be alleviated by using a new scheme proposed in this article [4] which we call the parallel sequential induction network (PSIN). PSIN takes advantage of both the parallel and sequential strategies in solving a problem. It consists of many nodes that would classify the incoming patterns into a few subcategories. Each node in PSIN is itself a neural network with one or more output neurons. It is important to notice that we do not expect a single node alone to accomplish the entire decision or classification job. It is the combination of many such nodes that would complement each other and in the end coorperatively get the job done.

2 The parallel sequential induction network

In a conventional neural network, training (or testing) patterns are presented to an input layer and the results read off from the output units. The adaptation of the connection weights are the consequence of all the training patterns because of the overall parallel strategy. However, it can be seen that this strategy may not be the best for problems represented by a multi-branched decision tree. An example is shown in *Fig.*1., where $a_1, a_2, a_3, ...$ are attributes to be tested at the corresponding nodes 1, 2, 3, ...

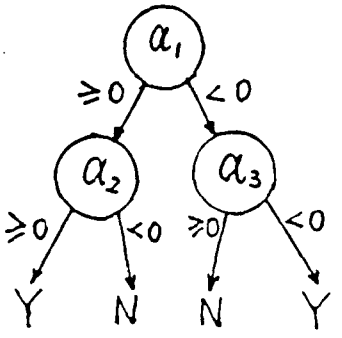


Fig. 1

Suppose there are only two pattern classes *yes* and *no*. For an input pattern, if its test result in node 1 is positive, it is channeled to the node 2 to test for attribute a_2. If the result is again positive we classify the pattern as *yes* and so on. We note that in order to classify a pattern, not all attributes are tested (or relevant). For some patterns a_1 and a_2 are the determinative attributes and a_3 is irrelevant. But for other patterns a_1 and a_3 are important and a_2 is irrelevant.

If we expect a three layered network to be able to extract automatically the three attributes a_1, a_2, and a_3, we are actually assuming that the input patterns are uniform in their regularity which is however not true. This problem would get worse dramatically as the depth of the decision tree is increased. The number of relevant attributes become much less than those irrelevant and their voting power in a decision process could be swamped easily by noise signals from those irrelervant attributes in an all parallel arrangement.

To remedy this problem, or to recognize that some attributes are important for a fraction of the patterns but not at all for others, we propose the parallel sequential induction network. The PSIN divides the task of classification of patterns into steps. The first step is to construct a node that extract the most important attributes for the largest fraction of patterns from all the input patterns. We use information entropy to measure the quality of this node. After training, we expect the node to do its best to classify the patterns. However, usually what the first node could accomplish is only to purify the patterns in each branches. We set a criterion for the purity. If the purity in a branch is lower than the criterion, we use the subset of patterns that were channeled into the branch by the first node to train another node. We again maximize the information gain and purify the classification of the subset of patterns. This procedure of branching and purification is continued until a satisfactory performance is achieved.

3 Training of nodes

The objective of a node is to purify the classification of the patterns that come to this node. The purity of the patterns are measured by an information function. Before entering the node, we assume there are $N^+ + N^- = N$ number of patterns. where N^+ is the number of *yes* patterns and N^- is the number of *no* patterns. The information entropy function (or impurity function) that characterizes the impurity of these patterns is given by

$$S_b = \frac{1}{2}\left(\frac{N^+}{N} log \frac{N^+}{N} + \frac{N^-}{N} log \frac{N^-}{N}\right) \tag{1}$$

using the Shannon-Weaver entropy. In practice, we approximate the above with a simpler function

$$S_b = \left(\frac{N^+}{N}\right)^2 + \left(\frac{N^-}{N}\right)^2. \tag{2}$$

A node is in principle a neural network by itself. It consists of an input layer to which the input patterns are presented and an output layer that may contain one or more output units. Since one of the strategies of PSIN is to relieve the burden of classification task on a single network to a series of smaller networks (or nodes), each node of PSIN may be constructed using the simple perceptron architecture instead of other more complex networks. However, we should note that the training of the connection weights in this perceptron-like node is not the error correction scheme.

A node consisting of n output neurons will channel the patterns into 2^n possible branches. The impurity or entropy function after the classification by the node is given by

$$S_a = \frac{1}{N} \sum_{j_1 j_2 \cdots j_n} \frac{\left(N^+_{j_1 j_2 \cdots j_n}\right)^2 + \left(N^-_{j_1 j_2 \cdots j_n}\right)^2}{N_{j_1 j_2 \cdots j_n}} \tag{3}$$

where j_i (= 0 or 1) is the quantized output of the i_{th} neuron, $N^+_{j_1 j_2 \cdots j_n}$ is the number of *yes* patterns channelled into the branch with i_{th} neuron having the quantized output $j_i (i = 1, 2, ..., n)$. The information gain or the increase of purity by the node is then

$$\triangle S = S_a - S_b. \tag{4}$$

Since the node cannot change S_b, the trainning can only alter S_a and we adapt the weights to maximize S_a.

It is easy to show that

$$\triangle S = -\frac{2}{N} \sum_{j_1 j_2 \cdots j_n} \left[\left(\frac{N^-_{j_1 j_2 \cdots j_n}}{N_{j_1 j_2 \cdots j_n}} \right)^2 \triangle N^+_{j_1 j_2 \cdots j_n} + \left(\frac{N^+_{j_1 j_2 \cdots j_n}}{N_{j_1 j_2 \cdots j_n}} \right)^2 \triangle N^-_{j_1 j_2 \cdots j_n} \right]. \tag{5}$$

Note also that

$$\triangle N^+_{j_1 j_2 \cdots j_n} = \sum_{r=1}^{N} A^r \left\{ \prod_{k=1}^{n} [j_k O^r_k + (1 - j_k)(1 - O^r_k)] \right\} \sum_{l=1}^{n} \frac{(2j_l - 1) \triangle O^r_l}{j_l O^r_l + (1 - j_l)(1 - O^r_l)} \tag{6}$$

and

$$\triangle N^-_{j_1 j_2 \cdots j_n} = \sum_{r=1}^{N} (1 - A^r) \left\{ \prod_{k=1}^{n} [j_k O^r_k + (1 - j_k)(1 - O^r_k)] \right\} \sum_{l=1}^{n} \frac{(2j_l - 1) \triangle O^r_l}{j_l O^r_l + (1 - j_l)(1 - O^r_l)} \tag{7}$$

where A^r = 0 or 1, representing *yes* or *no* respectively, is the true classification of the pattern r and O^r_l is the analog output of the l_{th} neuron for the pattern r. We have

$$O^r_l = \frac{1}{1 + \exp(-\sum_j W_{lj} I^r_j)} \tag{8}$$

and

$$\frac{\partial O_l^r}{\partial W_{ij}} = O_l^r(1 - O_l^r) I_j^r \delta_{li} \tag{9}$$

Therefore, a stochastic gradient descent algorithm can be constructed with

$$\begin{aligned}
\Delta W_{il} &= \eta \frac{\partial S}{\partial W_{il}} \\
&= \frac{2\eta}{N} \sum_{j_1 j_2 \cdots j_n} \left\{ [2A^r(\frac{N^+_{j_1 j_2 \cdots j_n}}{N_{j_1 j_2 \cdots j_n}}) - (\frac{N_{j_1 j_2 \cdots j_n}}{N_{j_1 j_2 \cdots j_n}})^2] \right. \\
&\quad \left. \times \prod_{k=1}^{n} [j_k O_k + (1 - j_k)(1 - O_k)] \frac{(2j_i - 1) O_i (1 - O_i)}{j_i O_i + (1 - j_i)(1 - O_i)} I_l \right\}
\end{aligned} \tag{10}$$

where η is the learning rate and the summation over training patterns is dropped.

4 Result

We constructed a few examples and tested them with the PSIN against the back propagation on a three layered net. PSIN had proved to be superior than back propagation in all these cases.

These examples use three attributes constructed from ten input units

$$\begin{aligned}
a_1 &= v_1 + 1.5v_2 + 0.5v_3 + 0.4v_4 + 0.3v_5 + v_6 - v_7 + 2v_8 - v_9 + 0.9v_{10} \\
a_2 &= 0.1v_1 - 0.2v_2 + 0.3v_3 - 0.4v_4 + 0.5v_5 - 0.5 \quad - 0.4v_7 - 0.3v_8 \\
&\quad +0.2v_9 - 0.1v_{10} \\
a_3 &= v_2 + v_4 + v_6 + v_8 + v_{10}
\end{aligned}$$

All these are analog units. The componets of patterns $v_j, j = 1$ to 10 are generated randomly and uniformly between -1 and 1. In learning these cases, we use 5000 randomly generated training patterns. After learning we use another set of 5000 novel patterns to test the result.

(i) **Second Order XOR Problem**

This problem is represented in the decision tree in *Fig.*2. Using a node with single output neuron, PSIN cannot improve the purity of this node with any significant amount. This is expected because we know that XOR problem cannot be solved by perceptron with a single output unit. However, if we use two output neurons in the

node, then immediately the PSIN learned the problem in only 12 seconds cpu time on a Cray-XMP with an accuracy of 99.85% out of 5000 novel testing patterns. On the other hand, using a three layered network with back propagation, we found the results depend significantly on the number of hidden neurons. Using two hidden neurons, the error is as high as 25.67% after a trainig of 10 seconds cpu time. The result of three hidden neurons do not show improvement. The error is still 25.98% after a 7.2 seconds training time. Increasing the hidden neuron number to 10, the error is reduced to 2.25% which is still 15 times worse than the PSIN result and the training time is 69.7 seconds or about seven times longer than the PSIN method.

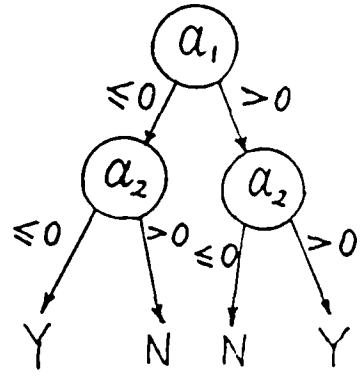


Fig. 2

(ii). **Third Order XOR**

This problem is represented in the decision tree in *Fig*.3. The PSIN approach starts from the premise that the problem may be solved using a node with a single output neuron. This being impossible, the program automatically increases the output neuron number in the node by one to try to improve its information gain. In the end, a node with three output neurons solved the problem with an accuracy of 98.9%. The cpu time including all the computations from one neuron node to three neuron node used is 39 seconds on Cray-XMP. If we knew before hand that we should use only three neuron node, then this time would be cut to at least a half. On the other hand, the automatic search for the right number of neurons to be included in a node is also one of the desirable feature of the PSIN approach

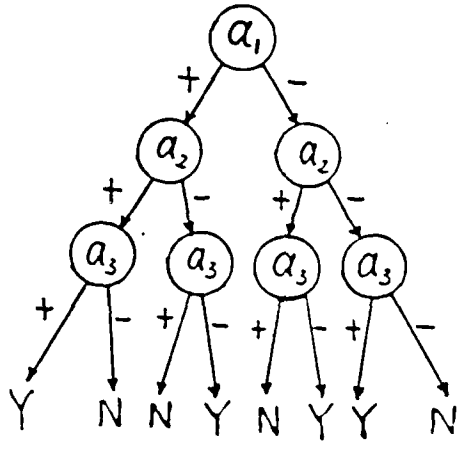


Fig. 3

The same problem run on a three layered network with back propagation has an error of 20% for three hidden neurons, 4.6% for six hidden neurons and the cpu

time spent is already 44.5 seconds. Increasing the hidden neurons to 10 results in a higher error rate of $5.5 \sim 7\%$ and the cpu time needed is around 70 seconds. We could not reduce further the error of back propagation scheme by including more hidden neurons.

(iii). **A Third Example**

As another example, we consider the decision tree in *Fig*.4. The PSIN approach automatically formd a two-node structure to solve this problem. The structure closely resembles the decision tree. The first node consists of one output neuron and the second consists of two output neurons. The network trained has an error rate of 0.75%. The time spent is 11.75 seconds. The back propagation scheme on a three layered net with three hidden neurons has an error of 7.45%. The time spent is 22.7 seconds.

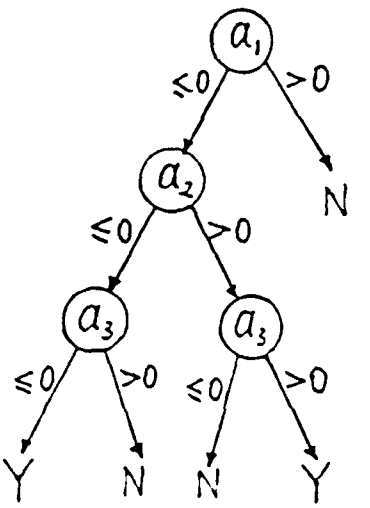


Fig. 4

5 Conclusion and discussion

In this paper, we have presented a scheme called Parallel Sequential Induction Network to construct automatically a tree of neural network nodes. Each node is trained to classify patterns channeled to it by a previous node. A stochastic gradient descent algorithm is presented to optimize the information gain (or the reduction of the impurity in the pattern sets) of the node. The PSIN scheme has been tested on a few decision tree problems and show a much superior result than the three layered network with hidden neurons and back propagation training.

We expect that for complex decision problems the combined parallel and sequential strategy would significantly alleviate the scale up problem for the all parallel multi-layered network. Since the scheme automatically search for the best organization of nodes and learned automatically the weights in each node to determine the important attributes, it can be very useful in many real life problems.

Acknowledgement

This work is supported by AFOSR and NSF.

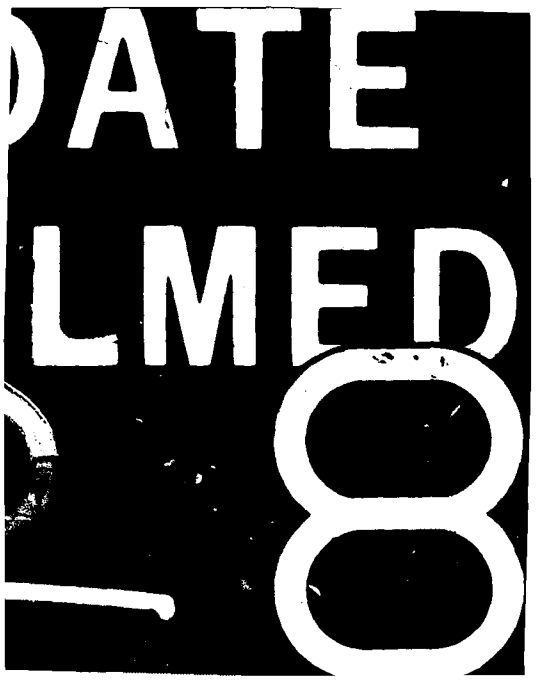
References

[1] L. Breiman, J. Friedman, R. Cohen, C.J. Stone, *Classification and Regression Trees*,Wadsworth Belmont, California(1984).

[2] J.R. Quinlan, *Induction of decision trees*, Machine Intelligence, Vol.1 p.81-106 (1986).

[3] D. Rummenlhart and J. McClelland, *Parallel Distributive Processing*, MIT Press(1986).

[4] G.Z. Sun, Y.C. Lee and H.H. Chen, *A Novel Net That Learns Sequential Decision Problems*, paper presented at IEEE Conference on Neural Infromation Processing System, Denver, Colorado(1987).


DATE
LMED
8