DTIC FILE COPY

AD-A230 663

CLASSIFICATION OF CORRELATION SIGNATURES OF SPREAD SPECTRUM SIGNALS USING NEURAL NETWORKS

THESIS

Richard A. Chapman

AFIT/GE/ENG/90D-11

DTIC ELECTE JAN 07 1991 S E D

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

91 1 3 142

AFIT/GE/ENG/90D-11

CLASSIFICATION OF CORRELATION SIGNATURES OF SPREAD SPECTRUM SIGNALS USING NEURAL NETWORKS

THESIS

Richard A. Chapman

AFIT/GE/ENG/90D-11

DTIC
ELECTE
JAN 07 1991
S D

Approved for public release; distribution unlimited

AFIT/GE/ENG/90D-11

CLASSIFICATION OF CORRELATION SIGNATURES OF SPREAD SPECTRUM SIGNALS USING NEURAL NETWORKS

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Electrical Engineering

Richard A. Chapman, B.E.E.

December 1990

Accession For
NTIS GRA&I
DTIC TAB
Unannounced
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or Special
A-1

Approved for public release; distribution unlimited

Preface

This research effort was driven by the need for a real-time method to determine features of an adversary's spread spectrum signals. The experiments of this thesis were designed to investigate the possibility of using neural networks in conjunction with a correlator to fill this need.

I would like to thank several people for their contributions to the development of this document. First, I thank my advisor, Dr. David M. Norman, for providing guidance and sharing knowledge throughout this research effort. Also, I am grateful to Daniel R. Zahirniak for his assistance in setting up and using his neural network software. I would also like to thank my committee members, Dr. Steven K. Rogers and Dr. Mark E. Oxley, for their advice for improving this thesis and Dr. Charles Garvin of Harry Diamond Labs for taking the time to generate the correlation signatures which made this research possible. Finally, I am deeply indebted to my wife, Brenda, and my children, Kimberly, Rockne, and Mitchell, for their encouragement, understanding, and support during the months in which this thesis was developed.

Richard A. Chapman

Table of Contents

		Page
Preface		ii
Table of Contents		iii
List of Figures		viii
List of Tables		ix
Abstract		xi
I.	Introduction	1-1
	1.1 Background	1-1
	1.2 Problem	1-1
	1.3 Summary of Current Knowledge	1-2
	1.4 Assumptions	1-2
	1.5 Objectives	1-3
	1.6 Scope	1-4
	1.7 Methodology	1-4
	1.8 Thesis Organization	1-5
II.	Background Material	2-1
	2.1 Introduction	2-1
	2.2 Spread Spectrum Signals	2-1
	2.2.1 Theory.	2-1
	2.2.2 ANN Connection.	2-2
	2.3 Artificial Neural Networks	2-2

Page

2.4 Multi-layer Perceptron Topology 2-3

2.5 Multi-layer Perceptron Training 2-5

2.5.1 Mean-Squared Error (MSE) Objective Function. 2-6

2.5.2 Cross Entropy (CE) Objective Function. . . . 2-6

2.5.3 Classification Figure of Merit (CFM) Objective Function. 2-7

2.5.4 Application of Objective Functions to ANN. . 2-7

2.5.5 Definition of Terms of Update Rules. 2-9

2.5.6 MSE, CE, and CFM Update Rules. 2-10

2.6 Multi-layer Perceptron Usage 2-14

2.7 Previous Research Using Perceptrons 2-14

2.7.1 Noise Reduction Using Perceptron. 2-14

2.7.2 Sonar Target Classification Using Perceptron. 2-15

2.7.3 Radar Data Classification Using Perceptron. . 2-15

2.7.4 Phoneme Recognition Using the MSE, CE, and CFM Functions. 2-16

2.8 Radial Basis Function (RBF) Networks 2-17

2.8.1 Nodes at Data Points Algorithm. 2-19

2.8.2 Kohonen Algorithm. 2-19

2.8.3 K-Means Algorithm. 2-20

2.8.4 Center at Class/Cluster Averages Algorithm. . 2-21

2.8.5 Set Sigmas to a Constant Rule. 2-23

2.8.6 Set Sigmas at P-Neighbor Averages Rule. . . . 2-23

2.8.7 Scale Sigmas by Constant Rule. 2-23

2.8.8 Back-Propagation Training. 2-24

2.8.9 Matrix Inversion. 2-25

2.8.10 Radial Basis Function Usage. 2-26

2.9 Previous Research Using RBFs 2-26

Page

2.9.1 Digit Classification Using RBFs. 2-26

2.9.2 Vowel Recognition Using RBFs. 2-27

2.9.3 Phoneme Labeling using RBFs. 2-27

2.10 Conclusion 2-28

III. Methodology 3-1

3.1 Introduction 3-1

3.2 Resources 3-1

3.2.1 Spread Spectrum Correlation Signatures. . . . 3-1

3.2.2 ANN Simulator Software. 3-2

3.2.3 Data Set Construction. 3-4

3.3 Notation and Definitions 3-3

3.4 Presentation of Network Results 3-8

3.4.1 Training Performance. 3-8

3.4.2 Test Performance. 3-9

3.5 Experiment Design 3-9

3.5.1 Run 1. 3-9

3.5.2 Run 2. 3-11

3.5.3 Run 3. 3-12

3.5.4 Run 4. 3-14

3.5.5 Direct Sequence Chip Rate Experiments. . . . 3-18

3.5.6 Frequency Hopped Hopping Rate Experiments. 3-19

3.6 Conclusion 3-21

IV. Results 4-1

4.1 Introduction 4-1

4.2 Run 1 - Two Classes Controlled Data Sets 4-1

4.2.1 Training Performance. 4-1

Page

4.2.2 Classification Accuracy on Test Vectors. . . . 4-2

4.2.3 Run 1 Summary. 4-2

4.3 Run 2 - 60/40% Training Class Mix 4-3

4.3.1 Training Performance. 4-3

4.3.2 Classification Accuracy on Test Vectors. . . . 4-3

4.3.3 Run 2 Summary. 4-4

4.4 Run 3 - Two Classes Randomly Selected Data Sets . . 4-6

4.4.1 Training Performance. 4-6

4.4.2 Classification Accuracy on Test Vectors. . . . 4-9

4.4.3 Nodes at Data Points Training Failure Alternatives. 4-9

4.4.4 Run 3 Summary. 4-11

4.5 Run 4 - Four Classes Randomly Selected Data Sets . . 4-13

4.5.1 Classification Accuracy on Test Vectors for Run 4. 4-15

4.5.2 Run 4 Summary. 4-15

4.6 Chip Rate Results. 4-16

4.6.1 Introduction. 4-16

4.6.2 Chip Rate Classification Results. 4-16

4.6.3 Chip Rate Experiments Summary. 4-18

4.7 Hopping Rate Experiment Results 4-19

4.7.1 Introduction. 4-19

4.7.2 Hopping Rate Classification Results. 4-19

4.7.3 Hopping Rate Experiments Summary. 4-21

4.8 Conclusion 4-22

Page

V. Conclusions and Recommendations 5-1

5.1 Conclusions 5-1

5.1.1 Two Class Network Performance. 5-1

5.1.2 Controlling Probability Matrix Symmetry. . . 5-1

5.1.3 Data Set Selection Method Effects. 5-1

5.1.4 Majority Vote Results. 5-2

5.1.5 Four Class Network Performance. 5-2

5.1.6 Classification Accuracy. 5-2

5.1.7 Training Times. 5-2

5.1.8 Chip and Hopping Rate Networks Performance. 5-3

5.2 Recommendations 5-3

Appendix A. Data Tables A-1

Appendix B. Data File Samples and Processing Software B-1

B.1 Preprocessing of Correlation Product Data Files. . . . B-1

B.2 Construction of Datasets. B-4

B.3 ANN Simulator Menu and Output Files. B-7

B.4 Processing of ANN Output. B-15

Bibliography BIB-1

Vita VITA-1

List of Figures

Figure		Page
2.1.	Three-layer Perceptron	2-4
2.2.	Single Perceptron Node	2-5
2.3.	Back-Propagation Network	2-8
2.4.	One Dimensional Radial Basis Function	2-17
2.5.	RBF Neural Network	2-18
3.1.	Diagram of a Majority Vote Network	3-5
3.2.	Run 4 Center at Class Averages - Seed 1	3-15
4.1.	Run R3MSE Training Performance	4-7
4.2.	Run R3aMSE Training Performance	4-8
4.3.	Run R3aCE Training Performance	4-8
4.4.	Run R3aCFM Training Performance	4-9
4.5.	Run 3 Center at Class Averages - Seed 6	4-11
4.6.	Run 4 CE Training Performance	4-14
B.1.	Direct Sequence Correlation Product CORR18 Before Processing	B-3
B.2.	Direct Sequence Correlation Product CORR18 After Processing	B-3

List of Tables

Table		Page
3.1.	Experiment Run Parameters	3-17
3.2.	Chip Rate Training Data Sets	3-18
3.3.	Chip Rate Networks Training Parameters	3-19
3.4.	Hopping Rate Training Data Sets	3-20
3.5.	Hopping Rate Networks Training Parameters	3-21
4.1.	Training Statistics for Run 1 Networks	4-1
4.2.	Output Summary Statistics for Distributions of Run 1	4-2
4.3.	Training Statistics for Run 2 Networks	4-4
4.4.	Output Summary Statistics for Distributions of Run 2	4-4
4.5.	Average Probability Matrices of Runs 1 and 2	4-5
4.6.	Training Statistics for Run 3 Networks	4-6
4.7.	Output Summary Statistics for Distributions of Run 3	4-10
4.8.	Training Statistics for Run 4 Networks	4-14
4.9.	Output Summary Statistics for Distributions of Run 4	4-15
4.10.	CR1 Test Vector Classification Results	4-16
4.11.	CR2 Test Vector Classification Results	4-17
4.12.	CR3 Test Vector Classification Results	4-17
4.13.	CR4 Test Vector Classification Results	4-18
4.14.	CR5 Test Vector Classification Results	4-18
4.15.	HR1 Test Vector Classification Results	4-20
4.16.	HR2 Test Vector Classification Results	4-20
4.17.	HR3 Test Vector Classification Results	4-21
4.18	HR4 Test Vector Classification Results	4-21

Table Page

A.1. Probability Matrices for Run 1 Center at Class Averages A-2

A.2. Probability Matrices for Run 1 Nodes at Data Points A-3

A.3. Probability Matrices for Run 2 Center at Class Averages A-4

A.4. Probability Matrices for Run 2 Nodes at Data Points A-5

A.5. Run R3MSE Training History Data A-6

A.6. Run R3aMSE Training History Data A-7

A.7. Run R3aCE Training History Data A-8

A.8. Run R3aCFM Training History Data A-9

A.9. Run 3 Center at Class Averages - Seed 6 . . . . . A-9

A.10.Probability Matrices for Run 3 Center at Class Averages A-10

A.11.Probability Matrices for Run 3 Nodes at Data Points A-11

A.12.Probability Matrices for R3MSE Networks A-12

A.13.Probability Matrices for R3aMSE Networks A-13

A.14.Probability Matrices for R3aCE Networks A-13

A.15.Probability Matrices for R3aCFM Networks A-14

A.16.Probability Matrices for Run 3 Majority Vote A-14

A.17.Run 4 Center at Class Averages - Seed 1 . . . . . A-15

A.18.Run 4 CE Training History Data A-15

A.19.Probabilities for Run 4 Center at Class Averages A-16

A.20.Probabilities for Run 4 Center at Class Averages A-16

A.21.Probabilities for Run 4 Cross Entropy A-17

A.22.Probabilities for Run 4 Cross Entropy A-17

Abstract

The major goals of this ~~research~~ thesis were to determine if Artificial Neural Networks (ANNs) could be trained to classify the correlation signatures of two classes of spread spectrum signals and four classes of spread spectrum signals. Also, the possibility of training an ANN to classify features of the signatures other than signal class was investigated. Radial Basis Function Networks and Back-Propagation Networks were used for the classification problems.

Correlation signatures of four types or classes were obtained from United States Army Harry Diamond Laboratories. The four types are as follows: direct sequence (DS), linearly-stepped frequency hopped (LSFH), randomly-driven frequency hopped (RDFH), and a hybrid of direct sequence and randomly-driven frequency hopped (HYB). These signatures were preprocessed and separated into various training and test data sets for presentation to the neural networks.

Radial Basis Function Networks and Back-Propagation Networks trained directly on two classes (DS and LSFH) and four classes (DS, LSFH, RDFH, and HYB) of correlation signatures. Classification accuracies ranged from 79% to 92% for the two class problem and from 70% to 76% for the four class problem. The Radial Basis Function Networks consistently produced classification accuracies from 5% to 10% higher than accuracies produced by the Back-Propagation Networks. The Radial Basis Function Networks produced this classification advantage in significantly less training time for all cases. In attempts to classify the signatures by parameters (e.g. chip rate of DS signatures and hopping rate of RDFH signatures) other than signal type or class, the results were inconclusive regarding the usefulness of ANNs.

CLASSIFICATION OF CORRELATION SIGNATURES OF SPREAD SPECTRUM SIGNALS USING NEURAL NETWORKS

I. Introduction

1.1 Background

Spread spectrum signals possess some very desirable qualities. The techniques for generating and decoding these signals make them difficult to jam or intercept (8:855). Due to these inherent benefits of spread spectrum, the United States can expect present and future adversaries to use spread spectrum techniques. Therefore, defeating spread spectrum is extremely important and desirable.

The U. S. Army Harry Diamond Laboratories has developed an acousto-optic correlator that can very effectively intercept and capture the correlation signatures of spread spectrum signals. Current methods for examining these captured signatures involve a human operator and are not practical for real-time investigation of the signatures during a conflict (1:1-1–1-2).

1.2 Problem

A method for examining the captured correlation signatures in order to identify features of an adversary's spread spectrum signals in real-time is needed. In conjunction with the correlator, a reliable method for extracting features of the signatures would be a major step toward defeating spread spectrum techniques.

1.3 Summary of Current Knowledge

The idea of using Artificial Neural Networks (ANNs) to examine the spread spectrum correlation signatures was suggested by researchers at Harry Diamond Laboratories (HDL). As a result, HDL is currently sponsoring AFIT research involving ANN classification of spread spectrum signals. An AFIT thesis by DeBerry served as the first step in this continuing research effort (1). The results of the previous research will be stated in the following paragraphs.

It was shown that an ANN (three-layer perceptron using the mean-squared error update rules) will train directly on the correlation signatures of a combination of direct sequence (DS) and linear-stepped frequency hopped (LSFH) spread spectrum signals. The perceptron networks correctly classified the test data from the two classes about 80% of the time after 10,000 training iterations. At 10,000 iterations, the networks had reached their maximum classification performance (1:5-1–5-2).

DeBerry also demonstrated that the perceptron's classification performances could be modeled as a probability matrix similar to those used to model communication channels. The symmetry of the matrix was shown to depend on the ratio of input vectors from the two classes (1:5-2–5-3).

Another result of DeBerry's thesis effort was that a majority vote of the three networks (trained on different, but equivalent signals and tested with the same signals) showed a slight improvement in classification performance over that of a single network. Also, the make-up of the training set was shown to have a much greater impact on network performance than either the presentation order of the training signals or the initialization of the network (1:5-4–5-5).

1.4 Assumptions

It will be assumed that the results are valid from the previous thesis effort using ANNs to classify spread spectrum signals. The assumption of validity is justified by

both the available documentation on the previous thesis effort and the thoroughness and competency with which the previous thesis committee followed and reviewed the research effort. Therefore, DeBerry's results will be used as a benchmark against which to compare the results of this thesis effort.

1.5 Objectives

The primary objectives of this thesis are to answer the following questions:

1. Can a Radial Basis Function (RBF) ANN be trained to classify DS and LSFH correlation signatures? If so, how does the classification performance of the RBF network compare to that of the three-layer perceptron ANN?
2. What classification accuracy can be expected when randomly selecting correlation signatures from the pool of available DS and LSFH to serve as training and test vectors? How does this accuracy compare with the accuracy produced by ANNs trained and tested with the vectors as selected for the previous thesis effort (1)?
3. Can the networks be trained to classify correlation signatures from the following four classes: direct sequence (DS), linear-stepped frequency hopped (LSFH), randomly-driven frequency hopped (RDFH), and a hybrid of direct sequence and randomly-driven frequency hopped (HYB). If so, how does the addition of the two classes affect the performance of the networks?
4. Can the classification accuracy of the perceptron or back-propagation networks be improved by taking a majority vote of three networks using three different learning algorithms (mean-squared error, cross entropy, and classification figure of merit) to update the link weights during training?
5. Can the back-propagation or RBF networks be trained to classify parameters of the correlation signatures other than the signal class?

1.6 Scope

The eventual goal of using ANNs to classify spread spectrum correlation signatures would be to develop and deploy hardware that could in real-time detect features of an adversary's spread spectrum signals (1:1-2). However, this research effort will be limited to a software analysis of the signatures in an effort to model the performance of an eventual hardware system.

As stated in the section above, the analysis will include attempts to train RBF and perceptron networks with correlation signatures from two and four classes. First, RBF networks will be trained with data sets containing the same correlation signatures as used in the previous thesis effort. Then, the networks will be trained with data selected at random from the set of signatures available for testing and training. These training attempts will be repeated for the four classes of signatures. If the perceptron networks can learn on the randomized data, a majority vote of networks using three types of back-propagation learning algorithms will be taken and analyzed. Finally, attempts will be made to train the networks on the chip rate of DS correlation signatures and the hopping rate of RDFH correlation signatures.

1.7 Methodology

The approach of this thesis effort will be very similar to that of the previous thesis effort for the two class problem. The HDL will transmit the acousto-optic correlation signature data files to an AFIT computer via MILNET. The data files will be pre-processed on a personal computer to the format required by the ANN simulator. The ANN simulator to be used in this effort was written by D. Zahirniak (17). For this effort, the software will be run on SUN 3 workstations. The ANNs will be trained with processed data files from the four classes of spread spectrum signals and then tested with different data files from the four classes. The guidelines for the research experiments are identified in a previous section of this chapter entitled Scope. The output classification performance of the ANNs will be processed on

a personal computer into a format suitable for presentation in the thesis and for analysis of the results.

1.8 Thesis Organization

Chapter 1 - has served as an introduction to the problem and sets the general guidelines for research.

Chapter 2 - will present background material for this research effort. The background will include information in the networks to be used and examples of recent research using these networks.

Chapter 3 - will contain the methodology for this research effort.

Chapter 4 - will present the results of the research effort.

Chapter 5 - will contain the conclusions drawn from the research results as well as recommendations for future related research.

II. Background Material

2.1 Introduction

The U. S. Army Harry Diamond Laboratories is currently sponsoring AFIT research in the area of classifying spread spectrum signals with Artificial Neural Networks (ANNs). This new and important area of research will be developed in this chapter by introducing the signals and networks to be used in the research and describing several recent research efforts using the networks.

2.2 Spread Spectrum Signals

Spread spectrum signals possess some very desirable properties. Their most important property is the advantage over interference which makes spread spectrum systems very difficult to jam (8:855). Since the United States can expect future adversaries to use spread spectrum, then defeating this technology is very important. The following sections contain an introduction to spread spectrum theory and an explanation of the reasoning behind the current research using ANNs to classify spread spectrum signals.

2.2.1 Theory. A good definition of spread spectrum is as follows:

> Spread spectrum is a means of transmission in which the signal occupies a bandwidth in excess of the minimum necessary to send the information; the band spread is accomplished by means of a code which is independent of the data, and a synchronized reception with the code at the receiver is used for despreading and subsequent data recovery. (8:855)

The common spread spectrum techniques include direct sequence (DS), frequency hopped (FH), and hybrid (DS/FH). The difference between the various techniques is the way a code sequence is used to spread the signal spectrum. For example, the

direct sequence digital code sequence modulates the carrier while the code sequence in a frequency hopping system is used to dictate carrier frequency shift increments. Regardless of the spread spectrum technique used, the code sequence is used to spread the bandwidth before transmission and is processed with the signal upon reception to remove the excess bandwidth and allow data recovery (6:1-3).

2.2.2 ANN Connection. The Harry Diamond Laboratories has developed an acousto-optic correlator that can very effectively intercept and capture the correlation signatures of spread spectrum signals. Current methods for examining these signatures involve a human operator and are not practical for real-time investigation of the signals during a conflict. The eventual goal of using ANNs to classify spread spectrum correlation signatures would be the hardware development and deployment of a network that could in real-time detect features of an adversary's spread spectrum signals. This implementation could prove crucial in a future conflict (1:1-1-1-2).

2.3 Artificial Neural Networks

Lippman's article "An Introduction to Computing with Neural Networks" provides an excellent introduction to ANNs. In general, neural networks consist of computational elements connected by weighted links. The weights are adjusted during the training and/or use of the network in an attempt to achieve human-like pattern recognition. Lippman reviewed six major network models by describing the design and purpose of each. Of these six ANN models, only two, the Kohonen Feature Map and the Perceptron, can be used with continuous valued input signals such as that provided to AFIT (3:4-6). DeBerry determined in his research that a three-layer perceptron or back-propagation network was a good choice for the spread spectrum classification problem (1).

However, in a 1989 article entitled "Pattern Classification Using Neural Networks", Lippman discusses a new type of ANN called Radial Basis Function (RBF)

classifiers. These networks have been compared to back-propagation (perceptron) networks for several speech classification problems. The RBF networks' classification performances were very similar to the back-propagation networks' performances. The RBF networks required significantly less training time to accomplish the similar performance (4:62).

Since this research effort will involve both the perceptron and RBF networks, the following review of ANNs will cover both the multi-layer perceptron using back-propagation training algorithms and the RBF network using a variety of training algorithms.

2.4 Multi-layer Perceptron Topology

The multi-layer perceptron consists of a set of input nodes, output nodes, and one or more layers of nodes in between. A three-layer perceptron, as shown in Figure 2.1, has two internal or hidden layers of nodes. The hidden layer nodes are non-linear computational elements that hold the key to the capabilities of the perceptron. As shown in Figure 2.2, the node of a multi-layer perceptron sums weighted inputs from every node in the previous layer of the network and passes this sum through a non-linearity such as a hard limiter or a sigmoid (3:4–15). An internal threshold is then subtracted from the value produced by the non-linearity. The resulting value is then passed via weighted links to every node on the next layer of the network (3:5–16).

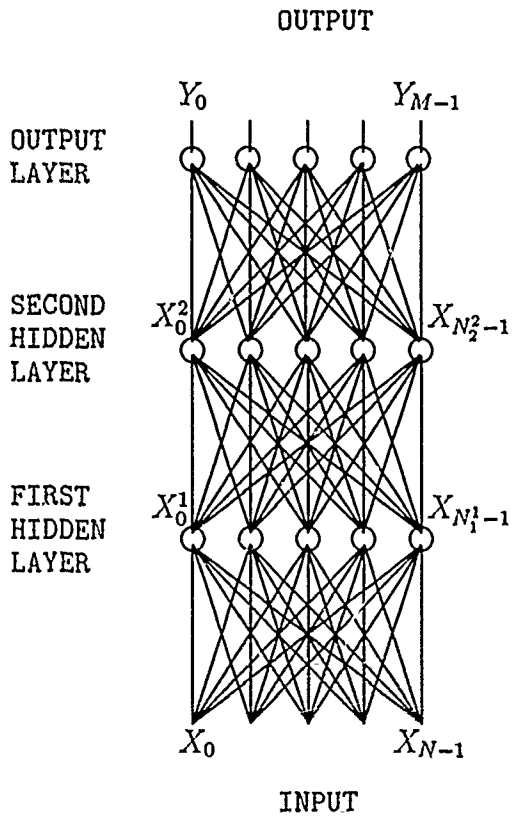

Figure 2.1. Three-layer Perceptron (3:16)

The output of a node can be expressed by the following functional relationship (3:5):

$$y = f\left(\sum_{i=0}^{N-1} x_i w_i - \theta\right) \tag{2.1}$$

where

y	=	output
w_i	=	i^{th} connection weight
x_i	=	i^{th} input
θ	=	threshold
f	=	nonlinear function

There exist no hard rules as to the number of hidden layer nodes required when using a multi-layer perceptron (13:57).

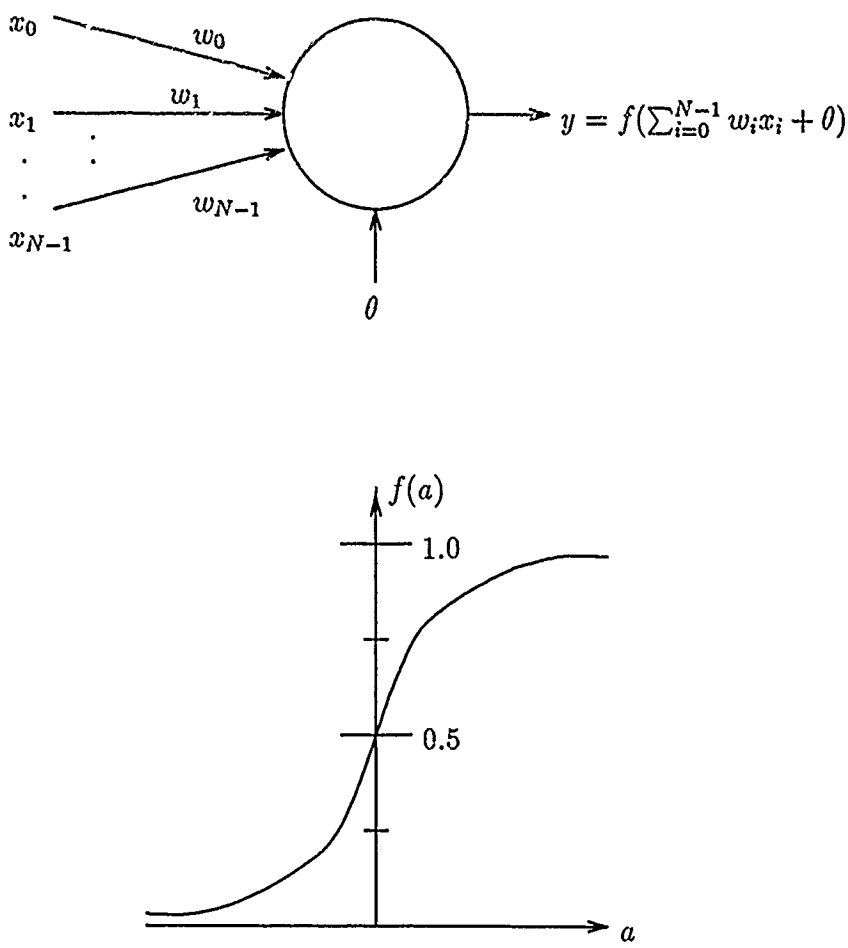

Figure 2.2. Single Perceptron Node (10:48)

2.5 Multi-layer Perceptron Training

The multi-layer perceptron is trained using a back-propagation training algorithm. For back-propagation algorithms, we shall use a sigmoid nonlinearity as shown in Figure 2.2. The sigmoid takes the following functional form:

$$f(\alpha) = \frac{1}{1 + e^{-\alpha}} \tag{2.2}$$

where α is the argument of the function in Equation (2.1). Before training the network, one sets all the link weights and node thresholds to small random values.

There are three classification objective functions currently used for back-propagation learning to adjust the link weights and node biases during training of the perceptron networks. These functions are the Mean Square Error (MSE), Cross Entropy (CE), and the Classification Figure of Merit (CFM) functions (15:217). Training is accomplished by presenting the continuous valued inputs and the desired output for each input to the network. As each training vector is presented to the network, the link weights and the node biases are adjusted based on the chosen classification function. These adjustments are the method by which the network learns the training data (11:322). The next three paragraphs will serve to introduce the three objective functions.

2.5.1 Mean-Squared Error (MSE) Objective Function. The MSE objective function used in back-propagation networks takes the following functional form:

$$MSE = \frac{1}{N}\sum_{n=1}^{N}(y_n - d_n)^2 \tag{2.3}$$

where N is the number of output nodes or classes, y_n is the network output for node n, and d_n is the desired output for node n. The MSE function acts to minimize the mean-squared error between the actual and desired outputs of the networks' output layer nodes (15:217). The desired output of the output layer node associated with the correct class is set to 1 and all other output nodes to 0 (3:17). The MSE function was the first of the three functions to be used with ANNs and is still the most widely used as evidenced by the number of papers published using this function.

2.5.2 Cross Entropy (CE) Objective Function. The CE objective function sees a node output as the probability that the node's desired output is a "1" which would make that node represent the correct class for a given input. The function

takes the following form:

$$CE = -\frac{1}{N}\sum_{n=1}^{N}[d_n \log(y_n) + (1 - d_n)\log(1 - y_n)] \tag{2.4}$$

where N is the number of output nc s or classes, y_n is the network output for node n, and d_n is the desired output for node n. The CE function acts to minimize the cross entropy between the actual and desired r.obability density functions driving a node (15:217). As with the MSE function, the desired output of the node representing the correct class is set to 1 and all other nodes' desired outputs are set to 0.

2.5.3 Classification Figure of Merit (CFM) Objective Function. The CFM objective function is as follows:

$$CFM = \frac{1}{N-1}\sum_{n=1,n\neq c}^{N}\left(\frac{\alpha}{1 + e^{(-\beta\delta_n+\zeta)}}\right) \tag{2.5}$$

where

δ_n = $y_c - y_n$
y_c = response of the correct node
y_n = response of the incorrect node
N = total number of output nodes or classes
α = sigmoid scaling parameter
β = sigmoid discontinuity parameter
ζ = sigmoid lateral shift parameter

The CFM function compares the output of the correct node that should be high with all other nodes that should be low. Then, a sigmoidal controlled by the parameters α, β, and ζ is applied to the differences. Thus, the CFM function concentrates on reducing misclassifications as a means to achieve a higher correct classification (15:219).

2.5.4 Application of Objective Functions to ANN. In order to apply a particular objective function to a network, update rules or equations based on the function

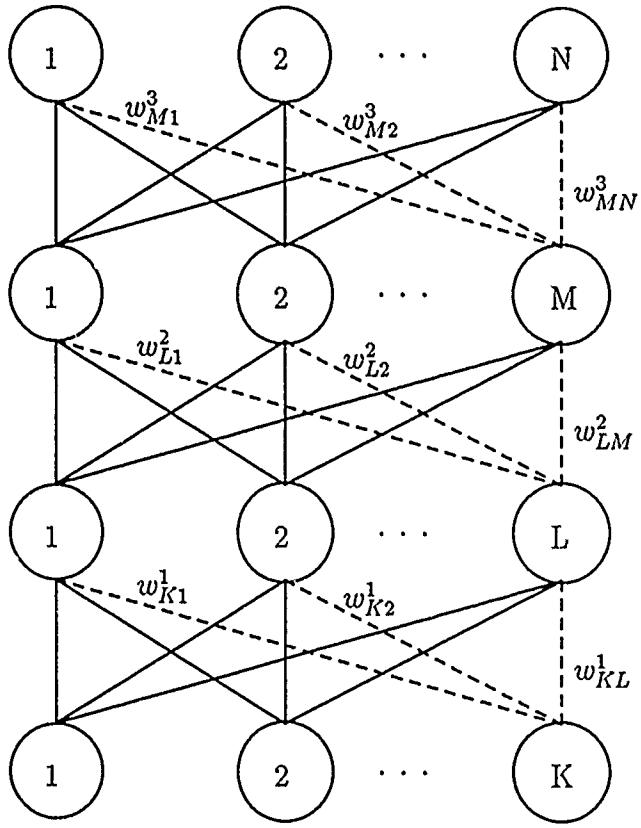

Figure 2.3. Back-Propagation Network

must be developed for the link weights and node offsets. The network will use these rules to learn the training data. These equations were derived for the MSE, CE, and CFM objective functions in (17). The MSE and CE update rules were derived by minimizing the MSE and CE functions with respect to network parameters. Therefore, the MSE and CE update rules act to minimize the error between the actual and desired network outputs. However, the CFM objective function is maximized with respect to the network parameters to derive the CFM update rules. The CFM update rules act to maximize the difference between the correct output node and all other output nodes (15:216).

2.5.5 Definition of Terms of Update Rules. This section of this document contains the update rules for the link weights and the node offsets of the MSE, CE, and CFM objective functions. The update rules are for the feed-forward ANN as shown in Figure 2.3. In the figure, the superscript on a weight identifies the network layer while the subscripts identify the nodes connected by each link. The following definitions should aide in understanding of the rules which are listed in the next subsection:

K = a specific input layer node as shown in Figure 2.3
L = a specific node in first hidden layer as shown in Figure 2.3
M = a specific node in second hidden layer as shown in Figure 2.3
N = a specific node in output layer as shown in Figure 2.3

w_{kl}^1 = weight linking a node k in layer 0 to a node l in layer 1
w_{lm}^2 = weight linking a node l in layer 1 to a node m in layer 2
w_{mn}^3 = weight linking a node m in layer 2 to a node n in layer 3

w_{KL}^1 = weight linking nodes K and L in Figure 2.1
w_{LM}^2 = weight linking nodes L and M in Figure 2.1
w_{MN}^3 = weight linking nodes M and N in Figure 2.1

θ_l^1 = node bias for w_{kl}^1
θ_m^2 = node bias for w_{lm}^2
θ_n^3 = node bias for w_{mn}^3

θ_L^1 = node bias for w_{KL}^1
θ_M^2 = node bias for w_{LM}^2
θ_N^3 = node bias for w_{MN}^3

y_n = actual output for a node n in output layer
d_n = desired output for a node n in output layer

y_N = actual output for output node N of Figure 2.1
d_N = desired output for output node N of Figure 2.1
y_C = correct node output for CFM function

η = learning rate
α = sigmoid scaling parameter for CFM function
β = sigmoid discontinuity parameter for CFM function
ζ = sigmoid lateral shift parameter for CFM function

$z_N = \frac{1}{1+e^{-\beta y_C + \beta y_N + \zeta}}$

2.5.6 MSE, CE, and CFM Update Rules. Each update rule listed below are for the back-propagation network as shown in Figure 2.3. The letter C is used to represent the correct output node in the CFM update equations, although node C is not shown in the figure. The networks adjust the link weights between adjacent layer nodes based on the weight update equations. The networks use the node bias equations to calculate the bias or threshold to be subtracted at each node as shown by Equation (2.1) in Section 2.4. The superscript (+) refers to the updated value of a weight or node bias after a training vector has been applied and the superscript (−) refers to the previous value. The equations are derived in (17).

For the MSE update rules, $\eta = 2A/N$ where A is a constant.
The MSE link weight update equations are as follows:

$$w_{mn}^{3+} = w_{mn}^{3-} - \eta(y_n - d_n)y_n(1-y_n)y_m \qquad (2.6)$$
$$\text{for each } m = 1,2,\ldots,M \text{ and } n = 1,2,\ldots,N$$

$$w_{lm}^{2+} = w_{lm}^{2-} - \eta \sum_{n=1}^{N}[(y_n - d_n)y_n(1-y_n)w_{mn}y_m(1-y_m)y_l] \qquad (2.7)$$
$$\text{for each } l = 1,2,\ldots,L \text{ and } m = 1,2,\ldots,M$$

$$w_{kl}^{1+} = w_{kl}^{1-} - \eta \sum_{n=1}^{N} \Big[(y_n - d_n) y_n (1 - y_n) \times \sum_{m=1}^{M} [w_{mn} y_m (1 - y_m) w_{lm} y_l (1 - y_l) y_k] \Big] \quad (2.8)$$
for each $k = 1, 2, \ldots, K$ and $l = 1, 2, \ldots, L$

The MSE node bias update equations are as follows:

$$\theta_n^{3+} = \theta_n^{3-} - \eta (y_n - d_n) y_n (1 - y_n) \quad (2.9)$$
for each $n = 1, 2, \ldots, N$

$$\theta_m^{2+} = \theta_m^{2-} - \eta \sum_{n=1}^{N} [(y_n - d_n) y_n (1 - y_n) w_{mn} y_m (1 - y_m)] \quad (2.10)$$
for each $m = 1, 2, \ldots, M$

$$\theta_l^{1+} = \theta_l^{1-} - \eta \sum_{n=1}^{N} \Big[(y_n - d_n) y_n (1 - y_n) \times \sum_{m=1}^{M} [w_{mn} y_m (1 - y_m) w_{lm} y_l (1 - y_l)] \Big] \quad (2.11)$$
for each $l = 1, 2, \ldots, L$

For the CE update rules, $\eta = A/[N ln(10)]$ where A is a constant.

The CE link weight update equations are as follows:

$$w_{mn}^{3+} = w_{mn}^{3-} + \eta (d_n - y_n) y_m \quad (2.12)$$
for each $m = 1, 2, \ldots, M$ and $n = 1, 2, \ldots, N$

$$w_{lm}^{2+} = w_{lm}^{2-} + \eta \sum_{n=1}^{N} [(d_n - y_n) w_{mn} y_m (1 - y_m) y_l] \quad (2.13)$$
for each $l = 1, 2, \ldots, L$ and $m = 1, 2, \ldots, M$

$$w_{kl}^{1+} = w_{kl}^{1-} + \eta \sum_{n=1}^{N} \Big[(d_n - y_n) \sum_{m=1}^{M} [w_{mn} y_m \times (1 - y_m) w_{lm} y_l (1 - y_l)] \Big] \quad \text{for each } k = 1, 2, \ldots, K \text{ and } l = 1, 2, \ldots, L \tag{2.14}$$

The CE node bias update equations are as follows:

$$\theta_n^{3+} = \theta_n^{3-} + \eta (d_n - y_n) \quad \text{for each } n = 1, 2, \ldots, N \tag{2.15}$$

$$\theta_m^{2+} = \theta_m^{2-} + \eta \sum_{n=1}^{N} [(d_n - y_n) w_{mn} y_m (1 - y_m)] \quad \text{for each } m = 1, 2, \ldots, M \tag{2.16}$$

$$\theta_l^{1+} = \theta_l^{1-} + \eta \sum_{n=1}^{N} \Big[(d_n - y_n) \sum_{m=1}^{M} [w_{mn} y_m \times (1 - y_m) w_{lm} y_l (1 - y_l)] \Big] \quad \text{for each } l = 1, 2, \ldots, L \tag{2.17}$$

For the CFM update rules, $\eta = A\alpha\beta/(N-1)$ where A is a constant.

The CFM link weight update equations are as follows:

For the weight linking a layer 2 node m to an incorrect output node n:

$$w_{mn}^{3+} = w_{mn}^{3-} - \eta z_n (1 - z_n) y_n (1 - y_n) y_m) \quad \text{for each } m = 1, 2, \ldots, M \text{ and } n = 1, 2, \ldots, N \tag{2.18}$$

For the weight linking a layer 2 node m to a correct output node C:

$$w_{mC}^{3+} = w_{mC}^{3-} + \eta \sum_{n=1, n \neq c}^{N} [z_n (1 - z_n) y_C (1 - y_C) y_m] \quad \text{for each } m = 1, 2, \ldots, M \tag{2.19}$$

$$w_{lm}^{2+} = w_{lm}^{2-} + \eta \sum_{n=1,n\neq c}^{N} [z_n(1-z_n)[y_c(1-y_c) \times w_{mc} - y_n(1-y_n)w_{mn}]y_m(1-y_m)y_l] \quad (2.20)$$

for each $l = 1, 2, \ldots, L$ and $m = 1, 2, \ldots, M$

$$w_{kl}^{1+} = w_{kl}^{1-} + \eta \sum_{n=1,n\neq c}^{N} \left[z_n(1-z_n)y_c(1-y_c) \sum_{m=1}^{M} \left[w_{mc} - y_n(1-y_n) \times \sum_{m=1}^{M} [w_{mn}y_m(1-y_m)w_{lm}y_l(1-y_l)y_k] \right] \right] \quad (2.21)$$

for each $k = 1, 2, \ldots, K$ and $l = 1, 2, \ldots, L$

For the node bias of an incorrect output node n:

$$\theta_n^{3+} = \theta_n^{3-} - \eta z_n(1-z_n)y_n(1-y_n) \quad (2.22)$$

for each $n = 1, 2, \ldots, N$

For the node bias of a correct output node C:

$$\theta_C^{3+} = \theta_C^{3-} + \eta \sum_{n=1,n\neq c}^{N} [z_n(1-z_n)y_C(1-y_C)] \quad (2.23)$$

$$\theta_m^{2+} = \theta_m^{2-} + \eta \sum_{n=1,n\neq c}^{N} [z_n(1-z_n)[y_c(1-y_c) \times w_{Mc} - y_n(1-y_n)w_{mn}]y_m(1-y_m)] \quad (2.24)$$

for each $m = 1, 2, \ldots, M$

$$\theta_l^{1+} = \theta_l^{1-} + \eta \sum_{n=1,n\neq c}^{N} \left[z_n(1-z_n)y_c(1-y_c) \sum_{m=1}^{M} \left[w_{mc} - y_n(1-y_n) \times \sum_{m=1}^{M} [w_{mn}y_m(1-y_m)w_{lm}y_l(1-y_l)] \right] \right] \quad (2.25)$$

for each $l - 1, 2, \ldots, L$

2.6 *Multi-layer Perceptron Usage*

The perceptron network is used by applying inputs of unknown types or classes to the network inputs. The network should be tested or used with different vectors than those it was trained with, although the vectors should be similar for the network to be of use. The network will choose a class which the input signal is most similar to from the classes presented during training (1:2-7).

2.7 *Previous Research Using Perceptrons*

A great deal of research has been accomplished over the last few years using the multi-layer perceptron. The following paragraphs will describe four such research efforts. In each case, multi-layer perceptrons using back-propagation learning rules were applied in an attempt to solve an existing problem.

2.7.1 Noise Reduction Using Perceptron. Tamura's experiments used the perceptron networks to reduce noise in speech signals. Other available noise reduction techniques have limitations due to the necessity of parameter estimates and other simplifying assumptions. A four-layer perceptron with 60 units per layer was chosen as a model that could in principle map any set of noisy signals to a set of noise-free signals. The MSE back-propagation algorithm was used with a sigmoid as the node non-linearity. The noisy speech was formed by mixing 5000 Japanese words with non-stationary computer room noise. Tamura chose 216 of these words to use in training and testing of the networks (12:553–554).

The training was accomplished by using the noisy words as inputs and the noise-free words as the target outputs. The network repeatedly scanned the words until convergence was achieved in about 200 scans and 3 weeks on an Alliant Supercomputer. The network was tested on the 216 words as well as other words and was shown to reduce noise. When the noise-reduced speech from the perceptron was

compared with that from the traditional power spectral method, the perceptron's speech was cleaner, although no more intelligible (12:554–556).

2.7.2 Sonar Target Classification Using Perceptron. Gorman used a two-layer perceptron to classify targets using sonar returns. The MSE back-propagation algorithm was used to train the network. The two targets were a metal cylinder and a rock shaped like a cylinder. The sonar returns were collected at various aspect angles. The experiments were designed to determine if the networks could classify the targets into two classes. Also, the effects that the number of hidden nodes and the aspect angles would have on classification performance were to be examined (2:1135–1137).

The experiments were performed with identical training and test vectors on networks containing 0, 2, 3, 6, 12, and 24 hidden nodes. Each of the networks were trained with both aspect angle dependent and independent returns. For the dependent case, returns for network training were selected to insure various aspect angles were represented. For the independent case, returns were selected at random (2:1137).

The results showed that the networks would converge with the sonar returns as inputs. The performance of the networks, in terms of percent of correct classification, improved as the number of hidden nodes increased from 0 to 12, although increasing the nodes to 24 produced no further improvement. In terms of aspect angle, the networks trained with aspect angle independent returns performed better than those trained with aspect angle dependent returns (2:1138–1140).

2.7.3 Radar Data Classification Using Perceptron. Another target recognition research effort was conducted by Troxel. He used a three-layer perceptron with MSE back-propagation training rules to classify radar data from tanks and trucks. The targets were positioned at various aspect angles. Troxel used a doppler segmenter on the radar returns and then transformed the segmented returns into a

position, scale, and rotation invariant (PSRI) feature space. The transformed data was correlated with the feature space and the peak of the correlation output was identified. Forty-nine points around the peak were chosen and normalized for presentation to the networks. The network performed with a classification accuracy of near 80% on the test data (14:593–600).

2.7.4 Phoneme Recognition Using the MSE, CE, and CFM Functions. Waibel compared the MSE, CE, and CFM functions for the /b, d, g/ phoneme recognition task. Japanese speech data was obtained, sampled, parsed of the phonemes and Hamming windowed. Then, 256-point DFTs were computed and used to generate coefficient spectra. The spectra were normalized for presentation to the back-propagation ANNs (15:216). Specific details concerning the experimental conditions can be found in (16).

The results showed that the test data error rate for the networks using the CFM function was 14% lower than the error rate using the MSE function and 18% lower than the rate when using the CE function. Also, due to the disjoint nature of the misclassified test vectors from the three networks, an arbitration scheme was developed to reduce misclassifications. The scheme involved summing of the different networks' outputs and dividing by the number of outputs summed. The highest of the arbitrated outputs was chosen as the correct class. Although summing the outputs of any two of the functions' networks showed some classification performance improvement, the best results were obtained when all three different networks were arbitrated. About 30% of the errors made by the MSE network alone were corrected when all three types of networks were arbitrated (15).

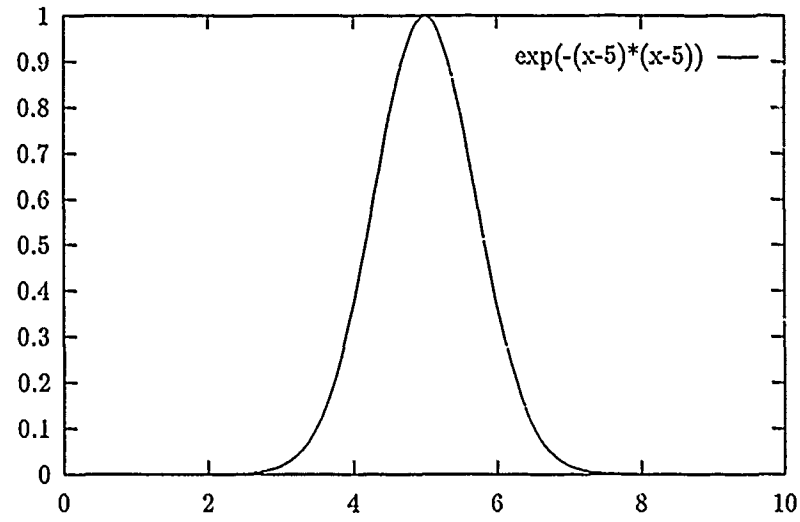

Figure 2.4. One Dimensional Radial Basis Function (17)

2.8 Radial Basis Function (RBF) Networks

An RBF is a radially symmetric function with a single maximum. A one dimensional RBF is shown in Figure 2.4. A RBF neural network acts to position the centers of the RBFs into regions where training vectors are present. An RBF network consists of three layers: an input layer to which the vectors are applied, a middle layer which places the RBFs, and an output layer with a linear function. All nodes of the input layer are connected to all nodes of the hidden, middle layer which are connected to all nodes of the output layer (5:133-135). A example of RBF network is shown in Figure 2.5. Weights link the input layer to the hidden layer and the hidden layer to the output layer. There are several algorithms for setting or adjusting both sets of weights. There are also several rules for determining the size or spread of the RBFs. The following sections will discuss some of the algorithms used to calculate the input link weights, the sigma rules used to determine the spread of the RBFs, and the methods for calculating the output link weights.

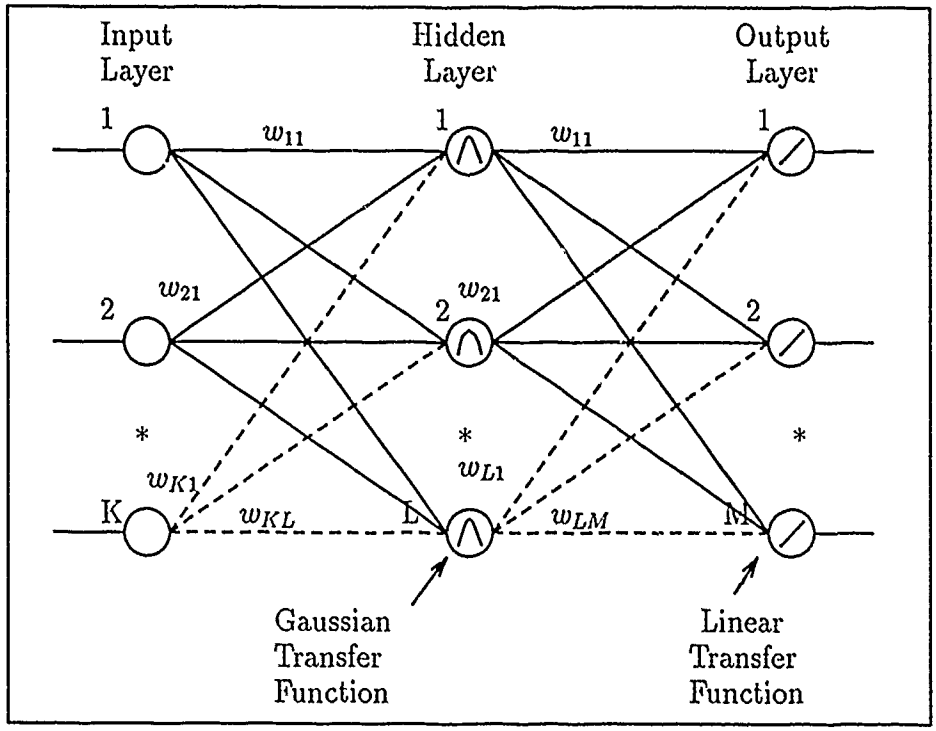

Figure 2.5. RBF Neural Network (17)

RBF networks differ from the back-propagation networks in several ways. First, the RBF always have one hidden layer of nodes while the back-propagation networks may have more than one. The RBF networks use gaussian-like functions and the back-propagation networks use sigmoids. The back-propagation networks generally require thousands of training iterations since all layers of link weights must be adjusted as function of the outputs. However, most RBF training algorithms may require one or very few iterations to position the functions. The training time required for training is usually many time less for RBF networks than for back-propagation networks. Previous research shows that RBF networks require more hidden layer nodes and training data than back-propagation networks for some classification problems (4:62) (5:133-134).

2.8.1 Nodes at Data Points Algorithm. This training algorithm calls for a RBF to be centered about a point corresponding to the features of each training vector. Therefore, a training set of 1,000 vectors would require 1,000 RBFs. The weight vector for the l^{th} RBF will be the same as the feature vector for the l^{th} training exemplar:

$$\bar{w}_l = \bar{x}_l \tag{2.26}$$

Therefore, the output for the l^{th} RBF due to the p^{th} input training vector is as follows:

$$y_l = e^{\sum_{k=1}^{K}\left(\frac{x_{pk}-x_{kl}}{2\sigma_{kl}}\right)^2} \tag{2.27}$$

where K is the number of input nodes or sample points per vector, x_{pk} is the value of the k^{th} sample point of input vector p, x_{kl} is the feature vector from input node to hidden layer node l, and σ_{kl} is the spread of the RBF of node l due to input node k.

Advantages of the Nodes at Data Points algorithm include the negligible time required for setting weights linking the input and hidden layers and that each RBF represents a class of data at it's peak output. Disadvantages include the large number of nodes which are required for large sets of training data. The time needed to compute the link weights between the hidden and output layers increases directly as the number of hidden layer nodes increases (17).

2.8.2 Kohonen Algorithm. The Kohonen Learning Algorithm is a clustering algorithm that learns the underlying probability density functions of the training vectors. For this algorithm, weights should be initialized to small random values and the vectors normalized to the range of the weights. The algorithm works with a rectangular grid of nodes the size of which must be selected by the network user. As each input is presented to the network, the Euclidean distance from that input to all

nodes of the hidden or Kohonen layer is computed based on the following equation:

$$d_j = \sum_{i=1}^{N}(x_i - w_{ij})^2 \tag{2.28}$$

where N is the number of inputs to the network, x_i is the input to the i^{th} node of the input layer, and w_{ij} is the weight between the i^{th} input node and the j^{th} hidden layer node. Then, the weights for the node with smallest distance and other nodes in the neighborhood of that node are updated by the following equation:

$$w_{ij}(t+1) = w_{ij}(t) + \eta(x_i - w_{ij}) \tag{2.29}$$

where w_{ij} is the weight between the i^{th} input node and the j^{th} hidden layer node and x_i is the input to the i^{th} node of the input layer, and η is the learning parameter or gain. As training proceeds for a specified number of iterations, the gain, η, and the size of the neighborhood decreases (10:64-67). When training is finished, RBFs would be positioned to represent a cluster of training vector, hopefully all from the same class.

An advantage of Kohonen training is that the number of nodes can be much less than the number of input vectors since one node of RBF can represent many vectors. A disadvantage is that Kohonen training takes much longer than other types of RBF training algorithms.

2.8.3 K-Means Algorithm. The K-Means algorithm adjusts the weights from the input layer to the hidden, middle layer in order to minimize a least mean square criterion. When this algorithm is used with an RBF network of K hidden layer nodes, RBFs are assigned to represent the first K training vectors. Then, each successive

training vector is assigned to a cluster as follows:

$$\bar{x}_p \in S_j \quad if \quad \|\bar{x}_p - \bar{w}_j\| < \|\bar{x}_p - \bar{w}_i\| \tag{2.30}$$

for $i = 1, 2, \ldots, K$ and $i \neq j$. For the above expression, $\bar{x}_p$ is the p^{th} pattern vector, S_j is the j^{th} cluster, $\bar{w}_j$ is the weight associated with S_j, and $\bar{w}_i$ is the weight associated with a different cluster S_i. The distances used are Euclidean distances. Once all the training vectors have been assigned to clusters, the average of each cluster is computed as follows:

$$\bar{w}_l = \frac{1}{N_c} \sum_{n_c=1}^{N_c} \bar{x}_{n_c} \tag{2.31}$$

where N_c is the number of training vectors assigned to the cluster, $\bar{x}$ is a pattern vector assigned to the cluster, and $\bar{w}_l$ is the weight associated with cluster l. Training continues until the weights, or cluster centers, no longer change when inputs are presented (17).

An advantage of this training algorithm is that the number of nodes can be significantly less than the number of training vectors. A disadvantage is that the number of nodes must be selected prior to training and there is no formula or method for determining the best number.

2.8.4 Center at Class/Cluster Averages Algorithm. With this algorithm, the weights or cluster centers are adjusted or shifted during training to points representing the centers of clusters from the same class. The first training vector and the class of that vector is presented to the network. A node is created with weights to match this vector. Then, the remaining training vectors are presented to the network and

clusters are assigned as follows:

$$\bar{x} \in S_j \quad if \quad \|\bar{x} - \bar{w}_j\| < \|\bar{x} - \bar{w}_i\| < C \tag{2.32}$$

and the class of $\bar{x}$ is the same as S_j. Here, $\bar{x}$ is the new pattern vector, S_j is the j^{th} cluster, $\bar{w}_j$ is the weight associated with S_j, $\bar{w}_i$ is the weight associated with S_i, and C is the selected cluster radius. If the distances to all present nodes of the same vector class as the new vector are greater than C, a new node or cluster center is created. If the new training vector is assigned to a present cluster, the weights of that node are adjusted as follows:

$$\bar{w}_j(t+1) = \bar{w}_j(t) + \frac{\bar{x}_{N+1} - \bar{w}_j(t)}{N+1} \tag{2.33}$$

where $\bar{w}_j(t)$ is the previous average of the N pattern vectors in that cluster, $\bar{w}_j(t+1)$ is the average after the addition of the new vector, and $\bar{x}_{N+1}$ is the pattern vector $N+1$ (17). Training continues until all training vectors are assigned to a cluster and the weights are adjusted for the last presented training vector.

Two advantages of the Center at Class Averages algorithm are that the number of nodes do not have to be pre-selected and these nodes may number much less than the number of training vectors. A disadvantage of this algorithm is that the radius C must be pre-selected with no method for determining the best radius.

2.8.5 Set Sigmas to a Constant Rule. For this rule, the sigmas of the RBFs are set to some constant, C. The output for an RBF node is calculated as follows:

$$y_{pl} = e^{-\frac{1}{2C}\sum_{k=1}^{K}(x_{pk}-w_{kl})^2} \tag{2.34}$$

where y_{pl} is the output for the l^{th} RBF due to the p^{th} training vector, x_{pk} is the value of the k^{th} point of input vector p, w_{lk} is the weight of RBF node l due to the k^{th} point of input vector p, and K is the number of input layers or sampled points (17).

2.8.6 Set Sigmas at P-Neighbor Averages Rule. For this sigma rule, the width of the RBF would be set equal to the root mean square of the Euclidean distances of the P nearest neighboring RBFs (5:137). The distance between two RBF nodes would be calculated from the following function:

$$d_{ij} = \sum_{k=1}^{K}(w_{kj} - w_{ki})^2 \tag{2.35}$$

where w_{kj} is the weight from input node k to node j, w_{ki} is the weight from input node k to node i, and K is the number of sample points in the input training vectors. The spread for the i^{th} RBF is set as follows:

$$\sigma_i = \sqrt{\frac{1}{P}\sum_{p=1}^{P} d_{ip}^2} \tag{2.36}$$

where P is the number of neighbors to be considered and d_{ip} is the distance between node i and node p (17).

2.8.7 Scale Sigmas by Constant Rule. In this algorithm, the sigma is preset to a constant and then decreased during training to prevent RBF nodes from responding

strongly to vectors from different classes. The output for each RBF node is calculated as follows:

$$y_{pl} = e^{-\frac{1}{2C}\sum_{k=1}^{K}(x_{pk}-w_{kl})^2} \tag{2.37}$$

where y_{pl} is the output of the l^{th} node due to input pattern p, C is the preset spread, K is the number of input layers or sampled points, x_{pk} is the value of the k^{th} sample point of input vector p, and w_{kl} is the weight of RBF node l due to the k^{th} point of input vector p. If this calculated $y_{pl} > T$, a preset threshold, then:

$$\sigma_l(t+1) = (1-C)\sigma_l(t) \tag{2.38}$$

where C is a scaling constant applied to the sigma until $y_{pl} < T$. This process is repeated for all training vectors (17).

2.8.8 Back-Propagation Training. The weights linking the hidden layer or RBF nodes to the output nodes can be found by any of the three back-propagation algorithms previously introduced. All three algorithm functions (MSE, CE, and CFM) would use their weight update rules to minimize the chosen function. The training for these weights would continue until the error rate reached an acceptable level. The training time of the networks will increase many times when back-propagation is used to train the link weights between the hidden layer nodes and the output nodes.

2.8.9 Matrix Inversion. The Matrix Inversion algorithm is a quick, effective way to calculate the link weights between the RBF and the output nodes. The total error due to all input vectors is as follows:

$$E = \frac{1}{2}\sum_{p=1}^{P}\sum_{m=1}^{M}(y_{pm} - d_{pm})^2 \tag{2.39}$$

where d_{pm} is the desired value for the m^{th} output node due to the p^{th} training vector, y_{pm} is the actual value, M is the number of output nodes, P is number of training vectors. For a particular node B in the hidden layer and a particular node D in the output layer, the error can be minimized by setting:

$$\partial E/\partial w_{BD} = 0 \tag{2.40}$$

Through a process of taking this derivative, defining several matrices, and manipulating the above equation and the matrices, the following weight equation can be derived (17):

$$w^*_{BD} = \sum_{l=1}^{L}\left(\sum_{p=1}^{P} y_{pl} d_{pD}\right) N_{Bl} \tag{2.41}$$

where

w^*_{BD}	=	optimized weight between nodes B and D
L	=	number of RBF or hidden layer nodes
P	=	number of training vectors
y_{pl}	=	output of the l^{th} RBF node due to p^{th} input vector
d_{pD}	=	desired output of the D^{th} output node due to the p^{th} input vector
N_{Bl}	=	$(M^T)^{-1}$
M	=	matrix containing the summation, over all vectors, of the product of each RBF output, for a given input pattern and the B^{th} RBF output for that pattern

2.8.10 Radial Basis Function Usage. To use an RBF network, an algorithm for both sets of weights linking the three layers and a sigma rule must be chosen. The input patterns are then presented to the network along with the desired outputs for the RBF algorithms except the Kohonen algorithm. Upon completion of training, vectors of an unknown class are presented to the network. Based on the position that the RBF network has placed the set of RBFs, the network chooses the class of the vector from the classes it was trained on.

2.9 Previous Research Using RBFs

Over the past several years, many research efforts have been accomplished using RBFs. The RBF networks have been applied to various classification tasks with success. The following subsections will describe three research efforts in which RBF networks were applied to classification problems.

2.9.1 Digit Classification Using RBFs. Nowlan used RBF networks to classify a set of hand drawn digits from twelve subjects. There were 320 training patterns and 160 test patterns. The patterns were digitized for presentation to the networks. Nowlan tried both spherical and ellipsoidal gaussians as the RBFs. RBFs were positioned in the input space by two algorithms. A form of the Nodes at Data Points algorithm was used in which the gaussians were assigned to points representing the training vectors with the highest probability of generating that observation. The K-means algorithm was also used in which all training patterns have an equal impact on the position of the RBFs (7:4-7).

The results of the experiment showed that networks trained with the K-means algorithm correctly classified the test data about 4% more accurately than networks trained using the Nodes at Data Points algorithm. The spherical RBFs performed about 2% better than the ellipsoidal RBFs. As the number of hidden layers nodes increased from 40 to 150, the classification accuracy increased by about 3%. The

best classification performance with an RBF networks was 94% using 150 spherical gaussians and the K-means algorithm. In much less CPU time, this result equaled the classification performance of a sophisticated back-propagation network (7

2.9.2 Vowel Recognition Using RBFs. Nowlan also applied a speaker independent vowel recognition tasks to RBF networks. The training and test vectors were digitized from the first and second formant frequencies of 10 vowels spoken by multiple male and female speakers. The networks were trained with 338 vectors and tested with 333 vectors. This data was applied to networks employing spherical RBFs and networks employing ellipsoidal RBFs. The two algorithms introduced in Subsection 2.9.1 were also applied to this problem (7:8-9).

The results again showed that the spherical gaussians provided a higher classification accuracy than did the ellipsoidal gaussians. Also, the RBF network with 100 gaussians out-performed the RBF network with 20 gaussians. As before, networks using the K-means algorithm provided better classification accuracy than networks using the hard algorithm. The best classification accuracy on the test data was 87%. A network with 100 spherical gaussians trained with a K-means algorithm provided this result. In a previous experiment, a two-layer back-propagation network had achieved a classification accuracy of only about 80.2% on the same data (7:9).

2.9.3 Phoneme Labeling using RBFs. For this experiment, the data consisted of vowel tokens segmented from a set of 98 sentences spoken by a single male speaker. The training set contained 758 tokens and the test set contained 759 tokens. There were 20 classes of tokens. The speech was sampled and then analyzed by either a Discrete Fourier Transform analysis or a 20^{th} order linear predictor analysis. The gaussians were placed with a Nodes at Data Points algorithm and the exemplar to receive the RBFs were selected at random. The RBFs' spreads were determined by the P-nearest neighbor rule with $P = 1$ or 2 (9:463-464).

The networks trained and tested with the speech analyzed with a linear predictor provided a slightly higher classification accuracy than the networks using the Fourier analyzed speech. Also, the classification accuracy of the RBF networks improved as the number of nodes was increased from 64 to 256. The best classification accuracy achieved for the test vectors was 73.3% which compared favorably to the best accuracy of a back-propagation network (73.0%). The RBF networks trained 2 to 3 times faster than did the back-propagation networks (9:464-465).

2.10 Conclusion

The classification of spread spectrum signal using ANNs is a very promising area of research. The desirability of defeating spread spectrum makes this research quite important. This importance as well as the successful results from previous research in the area mandate additional research efforts into the classification of spread spectrum signals using neural networks.

III. Methodology

3.1 Introduction

This chapter will provide the details of how the experiments of this thesis will be performed. First, the resources needed to conduct the research will be discussed. Then, the notation to be used in the rest of this document will be introduced. Finally, the design for each experiment will be detailed. Although all experiments will be introduced in this chapter before any results are reported in Chapter 4, results from previous runs did affect the design of subsequent runs in several cases.

3.2 Resources

This section will cover the resources used to perform the experiments of this thesis effort. These resources include the spread spectrum correlation signature data files, the ANN simulator software, and other software used to prepare the correlation signatures for presentation to the ANNs.

3.2.1 Spread Spectrum Correlation Signatures. The Harry Diamond Laboratories (HDL), sponsor of this thesis, provided the spread spectrum correlation signatures used to perform the experiments in this thesis. The signatures included four classes or types of spread spectrum signals: direct sequence (DS), frequency hopped stepped across frequency ranges by a linear stepper (LSFH), frequency hopped driven by a pseudo-random code (RDFH), and a combination of direct sequence and randomly-driven frequency hopped (HYB). Variations of each type of spread spectrum signal were simulated by varying parameters such as chip rate, hopping rate, pseudo-random code, etc. The signals were then fed into an acousto-optic correlator. The outputs of the correlator (correlation signatures) used for this thesis effort consisted of 1,000 data points in an ASCII file. Some of these files were

transmitted from HDL to an AFIT computer via MILNET using the file transfer protocol (FTP). The rest of the files were copied onto disks and mailed to AFIT.

3.2.2 ANN Simulator Software. The ANN simulator software was developed by D. Zahirniak (17). The software is written in the ANSI-C language and will be run on SUN 3 workstations. The software allows the user to choose from several types of radial basis function (RBF) networks and three types of back-propagation networks. Many of the available network algorithms and rules were described in Chapter 2. The user selects the type of network and specifies the run parameters from a menu file as shown in Appendix B. The results of the network execution are written to a file named by the user. The type of network chosen, the parameters specified, the percent correct classification, and the misclassified vectors can be found in this output file. In addition, a training and test history file detailing the network performance at every 1,000 iterations is created for back-propagation networks. An example of an output file and a history file can also be found in Appendix B.

3.2.3 Data Set Construction. The first step in this research effort was to transform the received correlation signatures into a form acceptable by the ANN simulator software. The received data files consisted of two columns of 1,000 numbers. The left column contained the sample numbers (1 to 1,000) while the right column contained the actual sampled data. These files were imported into LOTUS 1-2-3^{tm} where the left columns were deleted. The remaining data column was printed in an unformatted (to avoid page breaks) manner to files. These 1,000 point data files were imported into a software package called DADiSP Worksheet^{tm} where they were reduced to 500 points by averaging consecutive data point pairs. The peak of each 500 point signal was identified and 50 points around this peak were extracted. The 50 point signals were then normalized to values between +1 and -1. These normalized signals were written to files in the form of 50 ASCII numbers. A QuickBASIC^{tm} program converted these files into the exact format required by the

ANN simulator software. Specific details of the data set construction can be found in Appendix B.

3.3 Notation and Definitions

The following terms and definitions will be used in the remainder of this document:

class 1: Direct Sequence (DS) vectors

class 2: Linear Stepped Frequency Hopped (LSFH) vectors

class 3: Randomly Driven Frequency Hopped (RDFH) vectors

class 4: Hybrid (HYB) vectors

"good" classification: This classification criterium forces the network to choose one of the training classes for a given input. If the network's outputs for a given input were 0.45 for the class 1 output and 0.5 for the class 2 output, the network would choose class 2. All training and test percentages reported in this thesis are based on the "good" classification metric except the training history plots and tables for the back-propagation networks.

"right" classification: This criterium does not force the ANN to choose one of the training classes for a given input. For the ANN simulator software used for this thesis, the "right" metric requires a back-propagation network output to be 0.9 or higher for the network to choose the class represented by that output. The training history plots and tables reported in Chapter 4 for the back-propagation networks are based on the "right" classification metric. This metric is used during network training to force the ANN to learn the training data much closer than if the "good" metric was employed. The classification accuracy on the test vectors should be higher when the networks learn the training vectors closer.

P(good): The performance criteria used for the ANNs in this thesis is the probability of "good" classification on the test vectors. If a network yields a correct "good" classification for 75 of 100 test vectors, then the measured P(good) = 0.75.

$P(1 \mid 1)$: The probability of correctly classifying a class 1 test vector. If a network yields a correct "good" classification for 30 of 50 class 1 test vectors, then $P(1 \mid 1) = 0.60$.

$P(2 \mid 1)$: The probability of incorrectly classifying a class 1 test vector as a class 2 vector. $P(2 \mid 1) = 1 - P(1 \mid 1)$.

Run: A Run is distinguish from others runs by the make-up of the training and test vectors. The number of classes or the amount of control over which vectors are training and test exemplars are examples of characteristics that distinguish one run from another. The various Runs or data configurations used for the experiments in this thesis will be explained later in this chapter.

R1: The acronym for Run 1.

CA: The acronym for a RBF network using the Center at Class Averages training algorithm.

DP: The acronym for a RBF network using the Nodes at Data Points training algorithm.

R1CA: The acronym representing an RBF network using the Center at Class Averages training algorithm trained and tested with vectors of the Run 1 configuration.

R2DP: The acronym representing an RBF network using the Nodes at Data Points training algorithm trained and tested with vectors of the Run 2 configuration.

MSE: Earlier defined as the acronym for Mean Squared Error. It will also be used as an acronym for back-propagation networks using the Mean Squared Error weight and threshold update rules.

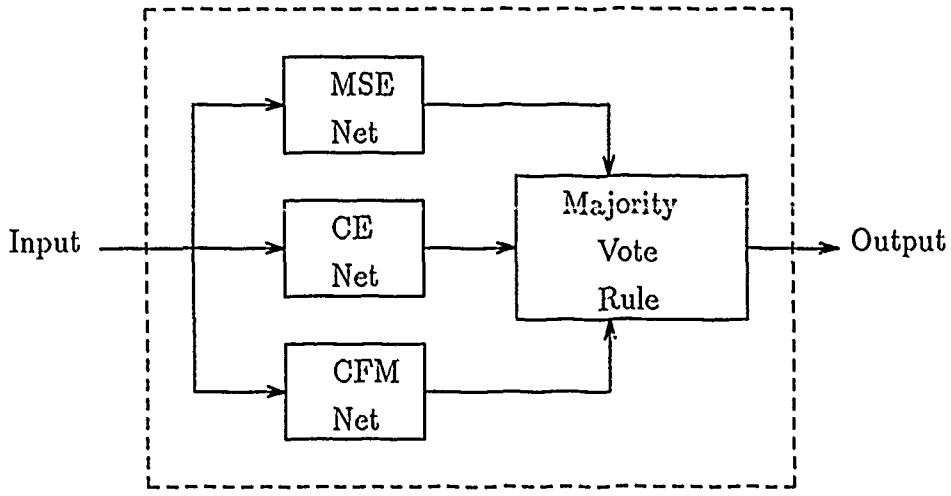

Figure 3.1. Diagram of a Majority Vote Network

CE: Earlier defined as the acronym for Cross Entropy. It will also be used as an acronym for back-propagation networks using the Cross Entropy weight and threshold update rules.

CFM: Earlier defined as the acronym for Classification Figure of Merit. It will also be used as an acronym for back-propagation networks using the Classification Figure of Merit weight and threshold update rules.

R3MSE: The Mean Squared Error weight and threshold update rules were used for a back-propagation network trained and tested with vectors of the R3 configuration.

R4CE: Th. Cross Entropy weight and threshold update rules were used for a back-propagation network trained and tested with vectors of the R4 configuration.

Majority Vote: For this thesis, majority vote will mean that the output of back-propagation networks using each of the three objective functions (MSE, CE, CFM) will be arbitrated as shown in Figure 3.1. If two or more of the three networks correctly classify a given input test vector, then the majority vote network will also correctly classify the given input vector.

MV: The acronym for the majority vote networks.

R3MV: The acronym representing a majority vote of the three types of back-propagation networks all of which were trained and tested with vectors of the R3 configuration.

The parameters that must be set for the RBF networks are as follows:

Nodes in Hidden Layer: For Nodes at Data Points networks, this must be selected. In general, placing a node or RBF at each training vector yields the best results. For Center at Class Averages networks, the number of nodes or RBFs used will be be allocated based on another parameter, the Average Threshold.

Average Threshold: This parameter sets the distance, between a presented input vector and the centers of the existing RBFs, required for creation of a new RBF or Cluster. The Average Threshold is only set for Center at Class Averages networks.

Sigma Threshold: This parameter sets the initial spreads of the RBFs or gaussians for both the Center at Class Averages networks and the Nodes at Data Points networks.

Output Threshold: This parameter sets the value of an existing gaussian or RBF that a presented input vector would have to reach for a new RBF to be created for a Nodes at Data Points network. For all experiments in this thesis, the Output Threshold will be set to 1, the peak of the gaussians, in order to assure a node or RBF will be centered at each training vector data point.

Sigma Factor: This parameter determines the amount by which the RBF spreads can be adjusted during training of networks using the Sigma Rule of Scale Sigmas by Constants.

Interference Threshold: This parameter is set to the amount of overlap of gaussians or RBFs that is required to cause a reduction in the spread of the gaussians.

The parameters that must be set for all three types of back-propagation networks to be used in this thesis are as follows:

Momentum: This parameter helps to control the rate at which a network converges. Although not shown in the objective function update rules of Section 2.5, a momentum term was implemented in the simulator software.

Iterations: This parameter is used to control the number of times that the training data set will be presented to the network.

Eta: This parameter is the learning rate, η, used in all weight and node bias update equations of Section 2.5.

The parameters that must be set for the MSE back-propagation networks are as follows:

Delta: This parameter controls the allowable difference between the desired correct output "1" and the actual value of that output during training. If this value is set to 0.1, the output node for the correct class must reach 0.9 for training to cease on that input training vector.

The parameters that must be set for the CE back-propagation networks are as follows:

Epsilon: This parameter controls the allowable difference between the desired correct output "1" and the actual value of that output during training. If this value is set to 0.1, the output node for the correct class must reach 0.9 for training to cease on that input training vector.

The parameters that must be set for the CFM back-propagation networks are as follows:

Alpha: This is the sigmoid scaling parameter, α, used in the CFM update rules of Section 2.5.6.

Beta: This is the discontinuity parameter, β, used in the CFM update rules of Section 2.5.6.

Zeta: This is the sigmoid lateral shift parameter, ζ, used in the CFM update rules of Section 2.5.6.

Delta: This parameter controls the minimum difference between the correct node's output value and all incorrect nodes' output values that should be achieved during training of the network. For this thesis, this parameter was set to 1 and the CFM networks' training was controlled by a software modification forcing the correct node to be a 0.9 or higher for the network' training history file to report a vector as being correctly classified.

3.4 Presentation of Network Results

This section will cover the methods to be used for reporting the performance of the networks used in Runs 1 through 4 of this thesis. One paragraph will cover the training performance data and another will cover the test performance data that will be reported in Chapter 4 of this thesis.

3.4.1 Training Performance. The training performance of the networks will be reported, although this performance will not be the main criteria used to determine the worth of the networks. For all networks, the P(good) achieved on the training vectors and the number of hidden layer nodes used will be reported. For the back-propagation networks, an average training performance curve will also be presented. The training performance at each 1,000 iterations, taken from the history

files generated by the simulator software, will be averaged for the set of networks to yield the training performance curve. The data used to generate training performance curves can be found in Appendix A.

3.4.2 Test Performance. The classification performance achieved on the test vectors by each type of network used for each run will be reported in the form of a table of summary statistics. As several of the definitions in the previous section suggest, the reported statistics will be in the form of probabilities. A previous thesis effort (1) has shown that the $\underline{P}$ matrix is a valid and useful way of presenting and evaluating an ANNs classification accuracy performance. Therefore, for each set of networks trained and tested for Runs 1 through 4 of this thesis, the overall probability of "good" classification (P(good)) and the conditional probabilities of "good" classification (e.g. $P(1 \mid 1)$) will be reported. The individual networks' classification performances from which the summary statistics were calculated can be found in tables in Appendix A.

3.5 Experiment Design

This section will explain the experiments to be performed in this thesis effort. This explanation will include the training and test vector set configuration used for each run, the purpose of each run, the types of networks trained and tested in each run, and the specific parameters selected for each network type. Table 3.1, located at the end of this section, contains a summary of the set-ups for each run.

3.5.1 Run 1. For Run 1 networks, the data will be set-up identical to the data in a run of the previous thesis effort in the area of classifying spread spectrum correlation signatures with ANNs (1). The training data will consist of 102 vectors and the test test data will consist of 100 vectors. For both the training and test data sets, half will be class 1 vectors and half class 2 vectors. Although the order of training vector presentation will be varied for each of the 30 networks to be trained,

the same vectors will be used for training and testing of the networks in each case. The data was selected for the training and test data sets based on a log file provided along with the correlation signatures by HDL. In general, the data set selection method used insures that each test vector is very similar to a training vector in signal parameters other than class (e.g. chip rate, PN code used, step-size, and step-width).

The purpose of Run 1 is to determine if RBF networks can be trained with a combination of DS and LSFH spread spectrum correlation signatures. If the networks can train on the data, the performance of the RBF ANNs will be compared to the back-propagation (MSE update rules) networks trained and tested for a previous AFIT thesis (1).

The networks to be trained for Run 1 will be two types of RBF networks. These types are the Center at Class Averages and the Nodes at Data Points networks. These two types of networks were chosen based on preliminary test results. The Center at Class Averages network was chosen because it offered a good classification performance in a small amount of time and with a small number of RBFs. The Nodes at Data Points network was chosen due to a classification performance unbeatable by other RBF networks tested. Other training algorithms tried in preliminary tests included the K-means and Kohonen algorithms. The K-means networks could not match, with a similar number of nodes and training time, the Center at Class Averages networks' classification accuracy. The Kohonen networks took many times longer to train than did the Nodes at Data Points networks and provided a lower classification accuracy.

There will be 30 Center at Class Averages networks and 30 Nodes at Data Points networks trained and tested with the Run 1 data configuration. The reported results for R1CA and R1DP will be the average and standard deviation of the classification accuracies of the individual networks. Both types of RBF networks will be run with data seeds 1 through 30. These seeds control the randomization of the

order of the vectors to presented to the ANNs. The selection of the specific parameters for each type of network was also determined based on preliminary testing. The parameters selected for the R1CA networks are as follows:

Average Threshold	=	2
Sigma Threshold	=	4
Output Layer Training	=	Matrix Inversion
Sigma Rule	=	Scale sigmas by a constant
Interference Threshold	=	0.4
Sigma Factor	=	0.1

The parameters for the R1DP networks are as follows:

Sigma Threshold	=	4
Output Threshold	=	1
Output Layer Training	=	Matrix Inversion
Sigma Rule	=	Scale sigmas by a constant
Interference Threshold	=	0.4
Sigma Factor	=	0.1

3.5.2 Run 2. For Run 2 networks, the data will also be configured in the manner as the data in a run of the previous thesis effort. The training data will consist of 85 vectors of which 60% or 51 will be class 1 vectors and 40% or 35 will be class 2 vectors. The training data was generated by randomly removing 17 class 2 training vectors from the 51 class 2 vectors used in training for Run 1. Since the output of 30 networks will again be averaged for Run 2, 30 different training vector sets were created by randomly removing 17 class 2 vectors 30 times. The test data will consist of the same 100 test vectors (50 from each class) that will be used for Run 1.

There are two purposes for Run 2. One purpose is to determine the impact that altering the ratio of the training vectors from the two classes will have on the symmetry of the $\underline{P}$ matrix. Also, the performance of the RBF networks of Run 2 can be compared to the classification performance of the back-propagation networks used in the previous thesis effort.

There will be 30 Center at Class Averages and 30 Nodes at Data Points networks trained and tested with the Run 2 data configuration. The reported results will be the average and standard deviation of the classificati- . performances of the individual networks. The data seeds used will be 121 through 150. The specific parameter choices for the R2CA and R2DP networks are identical to the choices for the R1CA and R1DP networks respectively.

3.5.3 Run 3. For Run 3 networks, the vectors selected to be training vectors and test vectors will be determined quite differently than the method used for the previous two runs. The data for Run 3 will be randomized. Although the same 202 vectors will be used as in Run 1, each vector will be a test vector for some networks and a training vector for some networks. From the 202 vectors (101 class 1 and 101 class 2), 51 class 1 and 51 class 2 vectors will be randomly selected as training vectors for each individual network. The 100 not chosen to be training vectors will serve as test vectors. The randomization or data seed will be changed for each network so that the make-up of the training and test vectors will be different for each network of a given type. The data selection method used for Run 3 should produce a more realistic classification performance than the method used for Run 1 in which detailed apriori knowledge was employed.

One purpose of Run 3 is to observe the networks' classification performances using training and test data sets chosen in a random manner. These performances will be compared with the classification performances achieved by the Run 1 networks to identify differences in classification accuracies produced by networks trained and tested with the different data set configurations. Also, as suggested by Waibel (15), an arbitration scheme will be employed to take into account the outputs of the MSE, CE, and CFM networks for a single decision. Therefore, a set of all three types of back-propagation networks (MSE, CE, CFM) will be trained with the Run 3 data configuration for use in an arbitration scheme. The arbitration scheme will involve taking a Majority Vote (MV) of the three networks' classifications to construct a

MV classification network. The MV classification performance will be compared to the performances of the three individual types of back-propagation networks.

For the RBF networks of Run 3, there will again be 30 Center at Class Averages and 30 Nodes at Data Points networks trained and tested. The reported results will be the average and standard deviation of the 30 individual networks. The data seeds used will be 1 through 30. Based on preliminary testing, the specific network parameters selected for R3CA and R3DP will once again be identical to the choices for R1CA and R1DP.

There will be 10 MSE back-propagation networks trained and tested with the Run 3 data configuration. These networks will be trained with data seeds 1 through 10. The initial link weights and node thresholds will be set to different random values for each of the 10 networks. The 10 R3MSE networks will contain 18 first layer hidden nodes and 10 second layer hidden nodes. The specific parameters chosen for the networks are as follows:

$$\begin{aligned} \text{Delta} &= 0.1 \\ \text{Momentum} &= 0.5 \\ \text{Eta} &= 0.3 \\ \text{Iterations} &= 50,000 \end{aligned}$$

For the majority vote scheme, 10 MSE, 10 CE, and 10 CFM networks will be trained and tested. The network designators for these back-propagation ANNs will be R3aMSE, R3aCE, and R3aCFM respectively. The 30 back-propagation networks will have 18 first hidden layer nodes and 10 second hidden layer nodes. The reported results will be the average and the standard deviation of the ten networks of each type. The data seeds used will be 1 through 10 for each type of network. The initial link weights and node thresholds will be set to the same random values for each of the 30 networks to be used for the majority vote decision rule. To obtain the majority vote classification performance, the MSE, CE, and CFM networks for data seed 1 will be arbitrated, for data seed 2 will be arbitrated, etc. Therefore,

the R3MV results will also be the average and standard deviation of 10 majority vote networks. The specific parameters for each type of back-propagation network were determined by preliminary testing. The parameters selected for the R3aMSE networks are:

Delta	=	0.1
Momentum	=	0.1
Eta	=	0.3
Iterations	=	50,000

The parameters selected for the R3aCE networks are:

Epsilon	=	0.05
Momentum	=	0.05
Eta	=	1.5
Iterations	=	30,000

The parameters selected for the R3aCFM networks are:

Alpha	=	1.0
Beta	=	4.0
Eta	=	0.14
Zeta	=	0.0
Delta	=	1.0
Momentum	=	0.1
Iterations	=	50,000

3.5.4 Run 4. For Run 4 networks, the data will be configured in the same manner as it was for Run 3. The only difference is the amount of data. The networks for Run 4 will be trained and tested with a combination of four classes of vectors instead of the two classes used in previous runs. There will be a total of 404 vectors (101 from each class) of which 204 will be training vectors and 200 will be test vectors. The vectors to be used for training each network will be randomly selected as were the vectors of Run 3. The reported results will again be the average and standard deviation for each set of network types.

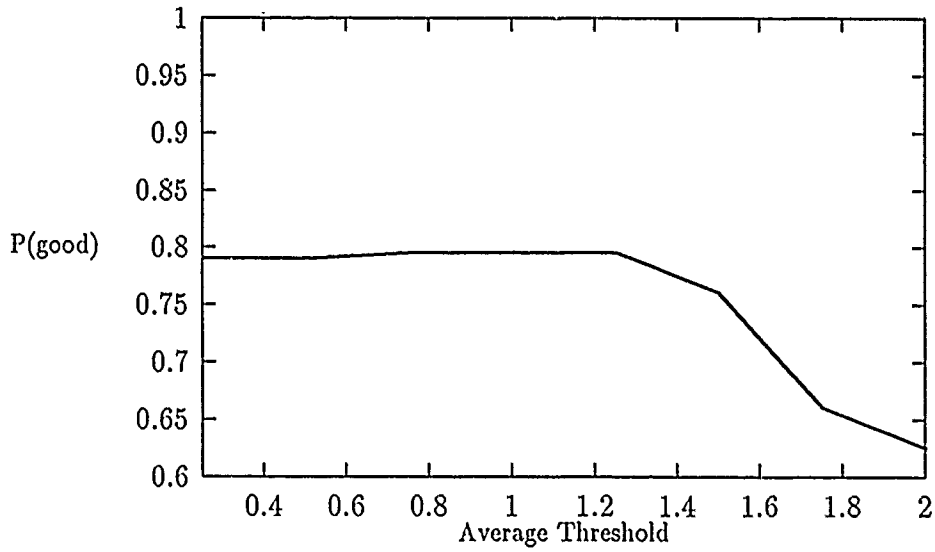

Figure 3.2. Run 4 Center at Class Averages - Seed 1

The purpose of Run 4 is to see if the networks will train on the four classes of spread spectrum correlation signatures. If the networks will train on the data, then the classification performance of the each type of network trained will be reported and compared to the classification performance of other networks trained and tested with the Run 4 data configuration.

For the RBF networks, preliminary testing was employed to determine the RBF training algorithm to use and the network parameters to select for the four class problem. Based on these tests, the Center at Class Averages networks with an average threshold of 1 was chosen. Figure 3.2 illustrates the effect that varying the average threshold can have on classification accuracy for the four class problem. The data used to generate the figure can be found in Table A.17. The preliminary tests showed that the networks would produce approximately 150 hidden layer nodes or RBFs with the average thresholds set to 1.

The network parameters chosen for the 30 Center at Class Averages networks are as follows:

Average Threshold	=	1
Sigma Threshold	=	4
Output Layer Training	=	Matrix Inversion
Sigma Rule	=	Scale sigmas by a constant
Interference Threshold	=	0.4
Sigma Factor	=	0.1

There will be ten CE back-propagation networks trained and tested with the Run 4 data configuration. The CE back-propagation networks with 24 first hidden layer nodes and 12 second hidden layer nodes were chosen based on the the classification performances achieved in preliminary testing. The choice for the remaining network parameters for the Run 4 CE networks are as follows:

Epsilon	=	0.05
Momentum	=	0.05
Eta	=	1.5
Iterations	=	50,000

Table 3.1. Experiment Run Parameters

Parameter	Run Designation			
	R1	R2	R3	R4
# of Nets Trained	60	60	100	40
# of Classes	2	2	2	4
Randomly Selected Training Exemplars	No	No	Yes	Yes
# of Training Exemplars	102	85	102	204
Training Set Mix class 1 ——> class 2 ——> class 3 ——> class 4 ——>	 50 % 50 %	 60 % 40 %	 50 % 50 %	 25 % 25 % 25 % 25 %
Identical Training Sets	Yes	No	Yes	Yes
# of Test Exemplars	100	100	100	200
Test Set percent per class	50 %	50 %	50 %	25 %
Identical Test Sets	Yes	Yes	No	No
Presentation Order	Variable	Variable	Variable	Variable
Majority Vote Networks	No	No	Yes	No
Distribution Nomenclatures	R1CA R1DP	R2CA R2DP	R3CA R3DP R3MSE R3aMSE R3aCE R3aCFM R3MV	R4CA R4CE

Table 3.2. Chip Rate Training Data Sets

Chip Rates	# of Vectors	Experiment Designation				
		CR1	CR2	CR3	CR4	CR5
1.0 MHz	33	16	10	11	18	00
1.5 MHz	24	12	10	00	00	18
2.0 MHz	11	06	10	00	00	00
2.5 MHz	22	11	10	00	18	18
3.0 MHz	11	06	10	11	00	00
Totals	101	51	50	22	36	36

3.5.5 Direct Sequence Chip Rate Experiments. If an ANN can be trained to classify spread spectrum correlation signatures based on the technique used to spread the signals, a next logical step would be to attempt to train ANNs to determine other features of an adversary's spread spectrum signals. To this end, ANNs will be trained on the chip rate of the DS correlation signatures received from the HDL. There will be 5 experiments or attempts to train the ANNs with various training data set mixes. Table 3.2 shows the chip rates of the 101 available DS signatures as well as the number of training vectors of each chip rate that will be used to train the 5 networks. All vectors not used for training the network of a given experiment will be used as test vectors for that experiment.

Experiments CR1 and CR2 are obviously designed to see if the networks will train on the five classes of chip rate. However, experiments CR3, CR4, and CR5 will be used to report the networks' classification of vectors with chip rates not used in training. If the networks can learn to classify based on the chip rates, a classification pattern should develop in which test vectors with chip rates not used in training are classified as having the nearest rate used in training. For example, a 3.0 MHz test vector in Run CR5 would be classified as a 2.5 MHz vector since 2.5 MHz is the closest rate used in training.

Based on previous runs' results and preliminary testing, the 5 experiments will use Nodes at Data Points networks with an RBF centered at each training vector.

Table 3.3. Chip Rate Networks Training Parameters

Network Parameters	Experiment Designation				
	CR1	CR2	CR3	CR4	CR5
Sigma Threshold	4.0	3.0	4.0	4.0	4.0
Output Threshold	1.0	1.0	1.0	1.0	1.0
Interference Threshold	0.3	0.3	0.3	0.5	0.3
Sigma Factor	0.1	0.1	0.1	0.1	0.1

Matrix Inversion will be used for training the weights between the hidden and output layers and Scale Sigmas by a Constant rule will be used to adjust the spreads of the RBFs or gaussians. Table 3.3 shows the specific parameters chosen for the networks.

3.5.6 Frequency Hopped Hopping Rate Experiments. Another logical step in the attempt to train ANNs to determine features of an adversary's spread spectrum signals would be to train the networks to classify the hopping rates of RDFH correlation signatures. To this end, there will be 4 experiments or attempts to train the ANNs on the hopping rates of the RDFH signatures obtained from the HDL. Table 3.4 shows the hopping rates of the 173 available RDFH signatures as well as the number of training vectors of each hopping rate that will be used to train the networks. All vectors not used for training the network of a given experiment will be used as test vectors for that experiment.

Experiments HR1 and HR2 are designed to see if the networks will train on the 8 classes of hopping rate. However, experiments HR3 and HR4 will be used to report the networks' classification of vectors with hopping rates not used in training. If the networks can learn to classify based on the hopping rates, a classification pattern should develop in which test vectors with hopping rates not used in training are

Table 3.4. Hopping Rate Training Data Sets

Hop Rates (hops/sec)	# of Vectors	Experiment Designation			
		HR1	HR2	HR3	HR4
62.5000	27	14	11	18	00
46.8750	27	14	11	00	24
39.0625	15	08	11	00	00
31.2500	18	09	11	00	00
15.6250	21	11	11	00	00
7.8125	27	14	11	18	24
6.2500	20	10	11	00	00
2.3438	18	09	11	00	00
Totals	173	89	88	36	48

classified as having the nearest rate used in training. For example, a 6.25 hops/sec test vector in Run CR3 would be classified as a 7.1825 hops/sec vector since 7.1825 hops/sec is the closest rate used in training.

Based on previous runs results and preliminary testing, the 4 experiments will use Nodes at Data Points networks with an RBF centered at each training vector. Matrix Inversion will be used for training the weights between the hidden and output layers and Scale Sigmas by a Constant rule will be used to adjust the spreads of the RBFs or gaussians. Table 3.5 shows the specific parameters chosen for the networks.

Table 3.5. Hopping Rate Networks Training Parameters

Network Parameters	Experiment Designation			
	HR1	HR2	HR3	HR4
Sigma Threshold	4.0	4.0	4.0	4.0
Output Threshold	1.0	1.0	1.0	1.0
Interference Threshold	0.2	0.6	0.3	0.95
Sigma Factor	0.1	0.1	0.1	0.1

3.6 Conclusion

The results of the experiments described in this chapter can be found in Chapter 4. The results will include the networks' classification statistics and the impact of these statistics on the stated purpose of each experiment.

IV. Results

4.1 Introduction

This chapter will contain the results of the experiments described in Chapter 3. For each type of network employed, the training and test performance will be reported. An analysis of the networks results as it pertains to the stated purpose of each run will also be included. As in the previous chapter, the networks' results will be organized by runs or data configurations. All output summary statistics tables presented in this chapter are excerpts of tables located in Appendix A.

4.2 Run 1 - Two Classes Controlled Data Sets

4.2.1 Training Performance. The networks for Run 1 successfully trained on the selected training vectors. The training performances of both the Center at Class Averages networks and the Nodes at Data Points networks can be found in Table 4.1. The number of nodes or RBFs match the number of training exemplars for the Nodes at Data Points networks while the number of nodes for the Center at Class Averages networks were determined by the networks based on the selected Average Threshold of 2. The Center at Class Averages networks trained in about 5 minutes CPU time and the Nodes at Data Points networks trained in about 15 minutes CPU time.

Table 4.1. Training Statistics for Run 1 Networks

Run ID	Statistic	# of Nodes	P(good)
R1CA	Mean	34.53	0.9229
	STD	2.29	0.0158
R1DP	Mean	102.00	1.0000
	STD	0.00	0.0000

Table 4.2. Output Summary Statistics for Distributions of Run 1

Run ID	Statistic	$P(1 \mid 1)$	$P(2 \mid 2)$	P(good)
R1CA	Mean	0.8347	0.9253	0.8800
	STD	0.0360	0.0216	0.0241
R1DP	Mean	0.9000	0.9400	0.9200
	STD	0.0000	0.0000	0.0000
R1MSE*	Mean	0.7380	0.9213	0.8297
	STD	0.0384	0.0560	0.0374

* Trained in a previous thesis effort

4.2.2 Classification Accuracy on Test Vectors. The classification accuracy the Run 1 networks achieved on the test vectors is shown in Table 4.2. While the Center at Class Averages classification accuracy is based on the average of 30 networks, the Nodes at Data Points is based on just 5 networks since the first 5 Nodes at Data Points networks (Data Seeds 1 through 5) performed identically. The MSE classification performance also shown in Table 4.2 is the result of 30 back-propagation networks trained using the MSE update rules.

4.2.3 Run 1 Summary. The results of the Run 1 networks show that RBF networks will train on a combination of DS and LSTH spread spectrum correlation signatures. The best classification performance was achieved using Nodes at Data Points networks with a hidden layer node or RBF placed at each of the 102 training vectors. The classification accuracy is identical from network to network when a node is placed at each data point. The 30 Center at Class Averages networks achieved a average classification accuracy 4% lower than the Nodes at Data Points networks. However, the Center at Class Averages networks trained in about one-third the time required to train the Nodes at Data Points networks. Also, the Nodes at Data Points networks utilized approximately three times as many hidden layer nodes as did the Center at Class Averages networks. Therefore, in this case, a trade-off exists

between increased training time and number of nodes on one hand and a decreased classification accuracy on the other hand.

The MSE back-propagation networks average classification accuracy on the test data was more than 5% lower than the Center at Class Averages RBF networks classification accuracy and more than 9% lower than the Nodes at Data Points RBF networks classification accuracy. A majority of the classification accuracy difference can be traced to the differences in classification accuracies on the class 1 DS signatures. Although no training time is available for the MSE networks, literature states that the training time for RBF networks is generally less than the training time for back-propagation networks (4) (5). Even if the training times were equal, the classification accuracy alone forces the conclusion that the RBF networks offer a substantial improvement over the MSE back-propagation networks for classifying the DS and LSFH spread spectrum correlation signatures as configured for Run 1 of this thesis.

4.3 Run 2 - 60/40% Training Class Mix

4.3.1 Training Performance. The training performances of the Center at Class Averages networks and the Nodes at Data Points networks are shown in Table 4.3. Since the number of nodes required for both types of networks were less for Run 2 than for Run 1, the training time required for both types of RBF networks was slightly less than the training times required for the same network types of Run 1. The Nodes at Data Point networks still required more than twice as long to train as did the Center at Class Averages networks.

4.3.2 Classification Accuracy on Test Vectors. The classification accuracy on test vectors achieved in Run 2 is shown in Table 4.4. The classification accuracies presented for both types of RBF networks are based on the average of 30 networks.

Table 4.3. Training Statistics for Run 2 Networks

Run ID	Statistic	# of Nodes	P(good)
R2CA	Mean	32.27	0.9341
	STD	1.46	0.0235
R2DP	Mean	85.00	1.0000
	STD	0.00	0.0000

The MSE classification performance also shown in Table 4.4 is the result of 30 back-propagation networks trained using the MSE update rules.

Table 4.4. Output Summary Statistics for Distributions of Run 2

Run ID	Statistic	$P(1 \mid 1)$	$P(2 \mid 2)$	P(good)
R2CA	Mean	0.8820	0.8273	0.8547
	STD	0.0384	0.0502	0.0300
R2DP	Mean	0.9220	0.8727	0.8973
	STD	0.0206	0.0384	0.0220
R2MSE*	Mean	0.7767	0.8320	0.8043
	STD	0.0485	0.0560	0.0370

* Trained in a previous thesis effort

4.3.3 Run 2 Summary. The results of the Run 2 networks show that the Nodes at Data Points networks again achieved about a 4% better classification accuracy on the test vectors than did the Center at Class Averages networks. This accuracy difference is consistent with the results of Run 1. Also, as in Run 1, the Nodes at Data Points networks required more time to train and used more hidden layer nodes than did the Center at Class Averages networks. The RBF networks again achieved a substantially higher classification accuracy than did the MSE back-propagation networks for the DS and LSFH data as configured for Run 2. As ex-

pected, the overall classification accuracy decreased for all three types of networks due to the removal of the 17 training vectors for each network.

The $\underline{P}$ matrix conditional probabilities were greatly changed by shifting the training data set class mix from 50% class 1 and 50% class 2 in Run 1 to 60% class 1 and 40% class 2 in Run 2. Table 4.5 contains the average conditional and overall probabilities achieved using Run 1 and 2 networks. All three types of ANNs produced a $\underline{P}$ matrix in Run 1 skewed in favor of $P(2 \mid 2)$. For Run 2, the RBF networks (R2CA and R2DP) produced $\underline{P}$ matrices skewed in favor of $P(1 \mid 1)$ while the MSE back-propagation networks produced a $\underline{P}$ matrix still skewed in favor of $P(2 \mid 2)$. Before the training classes were shifted in favor of class 1, the MSE matrix for Run 1 contained a difference in conditional probabilities ($P(2 \mid 2)$ - $P(1 \mid 1)$) more than twice as high as the differences of either type of RBF networks. This factor accounts for the difference in skew directions produced by the Run 2 networks. The results of Run 2 show that the $\underline{P}$ matrix symmetry can be controlled by adjusting the class mix of the training vectors for RBF networks as well as for the MSE back-propagation networks.

Table 4.5. Average Probability Matrices of Runs 1 and 2

Run ID	Statistic	$P(1 \mid 1)$	$P(2 \mid 2)$	P(good)
R1CA	Mean	0.8347	0.9253	0.8800
R2CA	Mean	0.8820	0.8273	0.8547
R1DP	Mean	0.9000	0.9400	0.9200
R2DP	Mean	0.9220	0.8727	0.8973
R1MSE*	Mean	0.7380	0.9213	0.8297
R2MSE*	Mean	0.7767	0.8320	0.8043

* Trained in a previous thesis effort

Table 4.6. Training Statistics for Run 3 Networks

Run ID	Statistic	# of Nodes	P(good)
R3CA	Mean	30.57	0.9267
	STD	2.65	0.0313
R3DP	Mean	102.00	0.9993
	STD	0.00	0.3774
R3MSE	Mean	18 - 10	1.0000
	STD	0.00	0.0000
R3aMSE	Mean	18 - 10	1.0000
	STD	0.00	0.0000
R3aCE	Mean	18 - 10	1.0000
	STD	0.00	0.0000
R3aCFM	Mean	18 - 10	0.8932
	STD	0.00	0.0575

4.4 Run 3 - Two Classes Randomly Selected Data Sets

4.4.1 Training Performance. The training statistics for the two Run 3 RBF networks (R3CA and R3DP) and the four Run 3 back-propagation networks (R3MSE, R3aMSE, R3aCE, and R3aCFM) are shown in Table 4.6.

For Run 3, the 30 Center at Class Averages RBF networks successfully trained on the randomly selected DS and LSFH signatures, but three of the 30 Nodes at Data Points networks would not train on the vectors selected based on the data seeds. For the three seeds, two or more selected training vector data points used to center the RBFs were too similar to use the Matrix Inversion algorithm for output layer training of the network. The training data sets forced the matrices used in the algorithm to become singular or near-singular which prevented the output layer training. Therefore, the training and test statistics provided for the Nodes at Data Points networks are based only on the 27 networks that did train. In Section 4.3.3, alternatives to the Nodes at Data Points Matrix Inversion problem will be discussed. As in Run 1, the Center at Class Averages networks trained in approximately 5

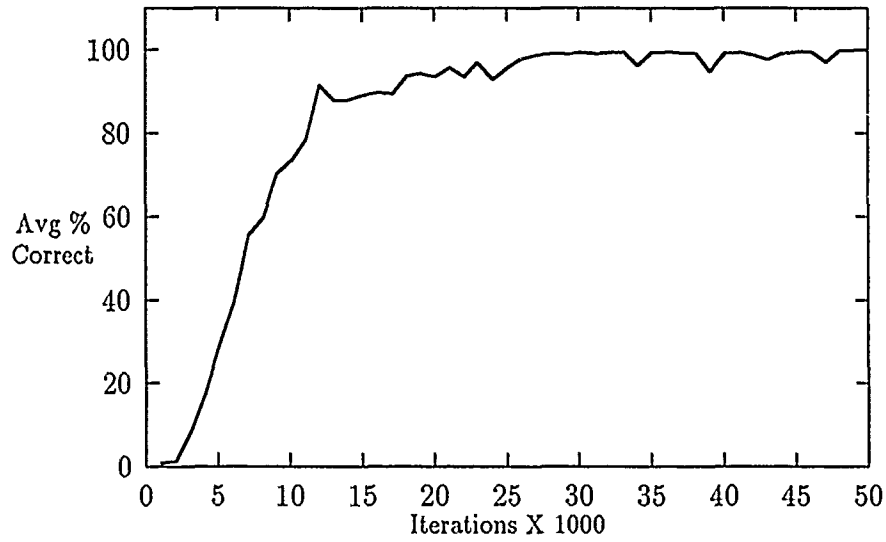

Figure 4.1. Run R3MSE Training Performance

minutes CPU time and the Nodes at Data Points networks trained in about 15 minutes CPU time.

For the four sets of back-propagation networks trained, the P(good) achieved on the training data is shown in Table 4.6. Also, plots of the training performances per 1,000 iterations are shown in Figures 4.1, 4.2, 4.3, and 4.4. The training plots are based on the "right" classification metric. The data from which the plots were produced can be found in Appendix A. The plots show that the R3aCE networks generally trained or converged in less iterations (about 10,000) than did the other sets of back-propagation networks. If training was terminated at 10,000 iterations, the R3aCE networks would train faster than the other networks. However, at 10,000 iterations, the CE networks would still require approximately an hour CPU time to train.

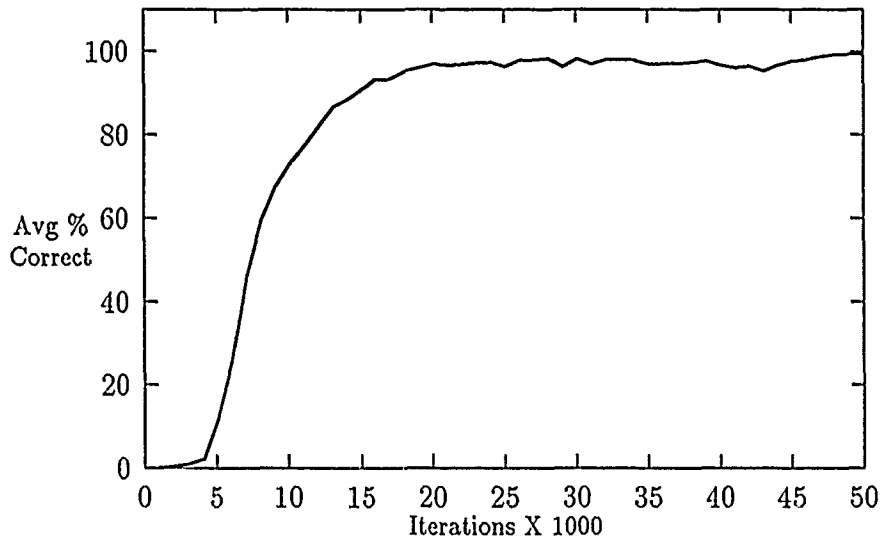

Figure 4.2. Run R3aMSE Training Performance

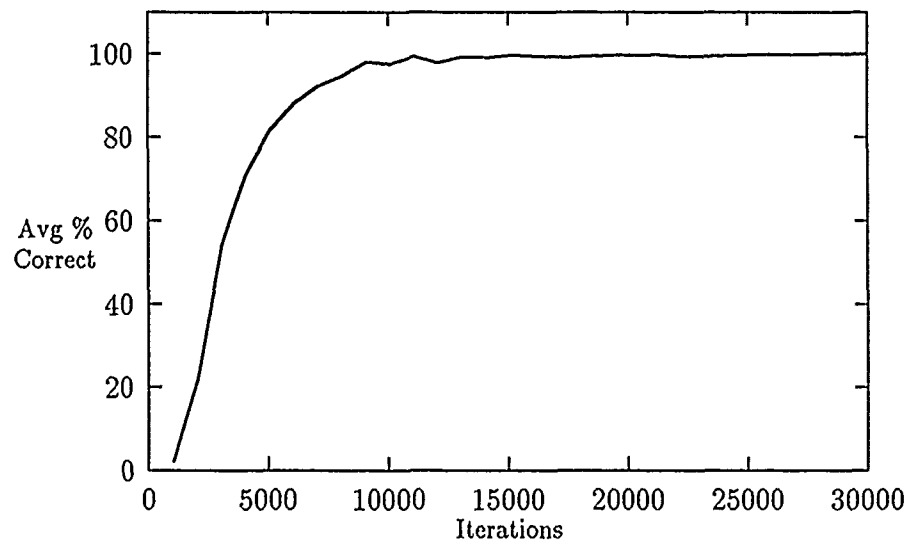

Figure 4.3. Run R3aCE Training Performance

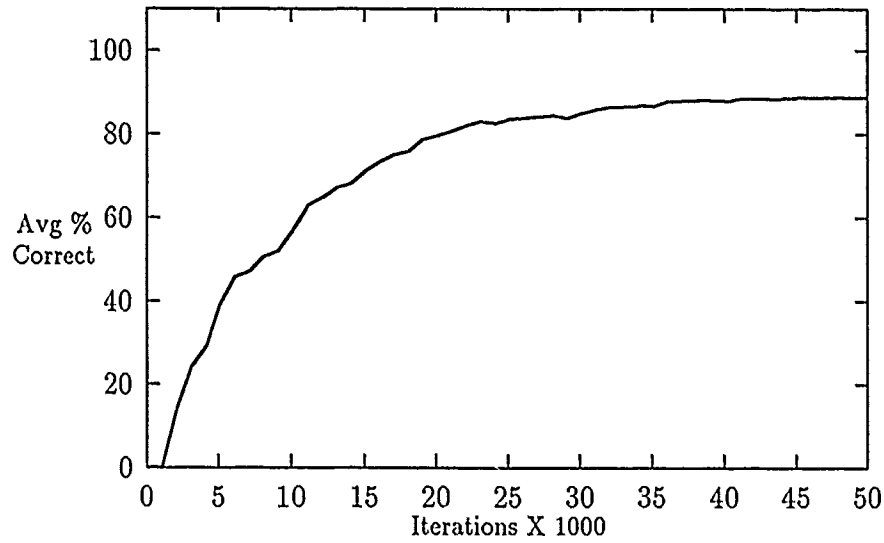

Figure 4.4. Run R3aCFM Training Performance

4.4.2 Classification Accuracy on Test Vectors. The average classification accuracy that the Run 3 networks achieved on the test vectors is shown in Table 4.7. The back-propagation networks' results are based on 10 networks each while the RBF networks' results are based on 30 and 27 networks as described in the previous section. The R3aMSE, R3aCE, and R3aCFM test vector classifications were arbitrated to produce the R3MV classifications.

4.4.3 Nodes at Data Points Training Failure Alternatives. Due to the fact that 3 out of 30 Nodes at Data Points networks would not train with the Run 3 data configuration, several possible methods for handling the problem will be introduced. The Nodes at Data Points networks, with a node or RBF centered at each data point, has provided the best classification accuracy for Runs 1 through 3. Therefore, work-arounds to the problem should focus on achieving a classification accuracy as close as possible to that achieved by the Nodes at Data Points networks.

Table 4.7. Output Summary Statistics for Distributions of Run 3

Run ID	Statistic	$P(1 \mid 1)$	$P(2 \mid 2)$	$P(\text{good})$
R3CA	Mean	0.8793	0.7567	0.8180
	STD	0.0680	0.0693	0.0400
R3DP	Mean	0.8807	0.8459	0.8633
	STD	0.0629	0.0546	0.0416
R3MSE	Mean	0.7760	0.8160	0.7960
	STD	0.0974	0.0587	0.0564
R3aMSE	Mean	0.8120	0.8240	0.8180
	STD	0.0895	0.0409	0.0522
R3aCE	Mean	0.8200	0.8040	0.8120
	STD	0.0869	0.0479	0.0479
R3aCFM	Mean	0.7720	0.6920	0.7320
	STD	0.1412	0.1455	0.0836
R3MV	Mean	0.8200	0.8020	0.8110
	STD	0.0827	0.0485	0.0504

Since only 3 of the 30 Nodes at Data Points networks did not train, a simple solution to the training failures can be found. The solution is to change the data seed and run another Nodes at Data Points network. The classification accuracy of the Nodes at Data Points networks would be achieved at the expense of increasing the training time. Since the Nodes at Data Points networks train approximately 4 times faster than any of the back-propagation networks, the classification accuracy and the training time advantage of the Nodes at Data Points networks over the back-propagation networks would almost certainly be preserved. Although this solution would work for this specific case, it would probably not be practical in the event of an actual deployment of a ANN.

A more general solution to the problem would be to substitute an RBF clustering training algorithm such as the Center at Class Averages or K-means algorithms for the Nodes at Data Points algorithm. Although the Center at Class Averages networks for Runs 1 through 3 of this thesis have been trained using substantially

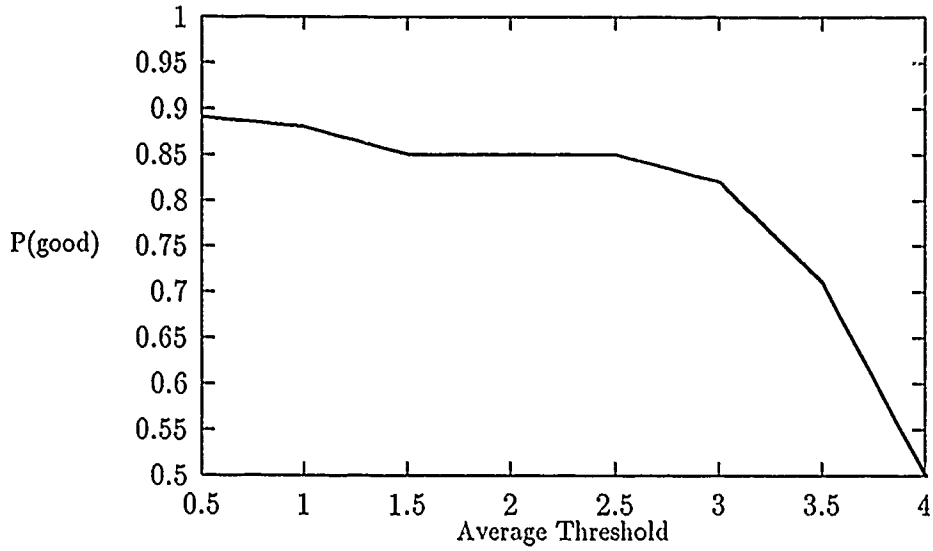

Figure 4.5. Run 3 Center at Class Averages - Seed 6

less nodes than the Nodes at Data Points networks, the average threshold can be adjusted to increase both the number of nodes or RBFs used and the classification accuracy. Figure 4.5 illustrates the effect on the classification accuracy when the average threshold is adjusted. This figure was produced with the Run 3 data configuration and all network parameters, except for the average threshold, set identical to parameters of the 30 Center at Class Averages networks trained for Run 3. With an average threshold of 0.05, the Center at Class Averages produces a P(good) of 0.89 which is the same as the Nodes as Data Points produced with the same data seed. The Center at Class Averages network generated 90 nodes or RBFs. The clustering method of the Center at Class Averages algorithm would force any training data points similar enough to prevent network training into the same cluster. Therefore, this algorithm should work for all seeds.

4.4.4 Run 3 Summary. The results of Run 3 again showed that the Nodes at Data Points networks (excluding the three that would not train) produced the

best classification accuracy of the networks trained. The Center at Class Averages, MSE, and CE networks produced classification accuracies of about 80% or about 5% lower than the Nodes at Data Points accuracy. For the Run 3 data configuration, the CFM produced a low classification accuracy as compared to the other types of networks.

The differences in the overall classification accuracy and the conditional probabilities between the Run 3 RBF networks and the Run 1 RBF networks are significant. The Nodes at Data Points networks and the Center at Class Averages networks of Run 3 produced overall classification accuracies of about 6% lower than the same types of networks had produced in Run 1. The MSE back-propagation networks produced similar classification accuracies for the data configurations of Runs 1 and 3. For the RBF networks, the Run 1 data configuration proved to contain test vectors significantly better represented by training data than did the Run 3 data configuration. Since the Run 1 data sets were selected to contain test vectors with similar parameters (e.g. chip rate) to the training vectors, the difference in classication accuracies achieved using the RBF networks in Runs 1 and 3 suggest that these networks may be useful in classifying signal parameters other than class. Also, the Run 3 networks produced $\underline{P}$ matrices that were either skewed in favor of $P(1 \mid 1)$ or very nearly symmetric whereas the Run 1 $\underline{P}$ matrices were heavily skewed in favor of $P(2 \mid 2)$. The $\underline{P}$ matrix differences serves as further evidence that the Run 1 data set configuration originally used in the previous thesis effort is very different from the data sets produced when training and test vectors were randomly selected from the pool of 202 DS and LSFH vectors.

Finally, the classification accuracy produced by the majority vote networks showed no classification accuracy advantage over the MSE and CE networks used in the majority vote scheme. Although the majority vote arbitration scheme is different from the arbitration scheme used for the phoneme recognition problem reviewed in Section 2.7.4, the majority vote scheme may have also produced classification

accuracy improvement over the individual network types under two conditions: the CFM networks classification performance on the Run 3 data configuration had been higher and the three types of networks had produced more disjoint misclassified test vector sets. For the phoneme classification problem, the CFM networks produced a slightly higher classification accuracy than did the MSE or CE networks and the three objective functions produced largely disjoint misclassified data sets (15). For the Run 3 data configuration of this thesis, the CFM networks produced a classification accuracy substantially lower than did the MSE and CE networks and the misclassified vector sets of the MSE and CE networks were very similar.

4.5 Run 4 - Four Classes Randomly Selected Data Sets

The training statistics for the Run 4 Center at Class Averages networks and the CE back-propagation networks are shown in Table 4.8. The 30 Center at Class Averages networks used an average of almost 150 nodes or RBFs and trained in under 30 minutes CPU time. The 10 CE networks were each trained for 50,000 iterations which took approximately 5 hours CPU time per network. Figure 4.6 is a plot of the average training performance of the Run 4 CE networks at each 1,000 iterations. The CE networks did not train to a steady state despite numerous runs with various networks parameters in an unsuccessful attempt to achieve such a state. Several of CE networks with the parameters as chosen for Run 4 were trained to 100,000 iterations. Despite doubling the training time, the networks had still not converged to a steady state and the classification accuracy on the test vectors decreased slightly over the additional iterations.

Table 4.8. Training Statistics for Run 4 Networks

Run ID	Statistic	# of Nodes	P(good)
R4CA	Mean	146.40	1.0000
	STD	4.33	0.0000
R4CE	Mean	24 - 12	0.9951
	STD	0.00	0.0061

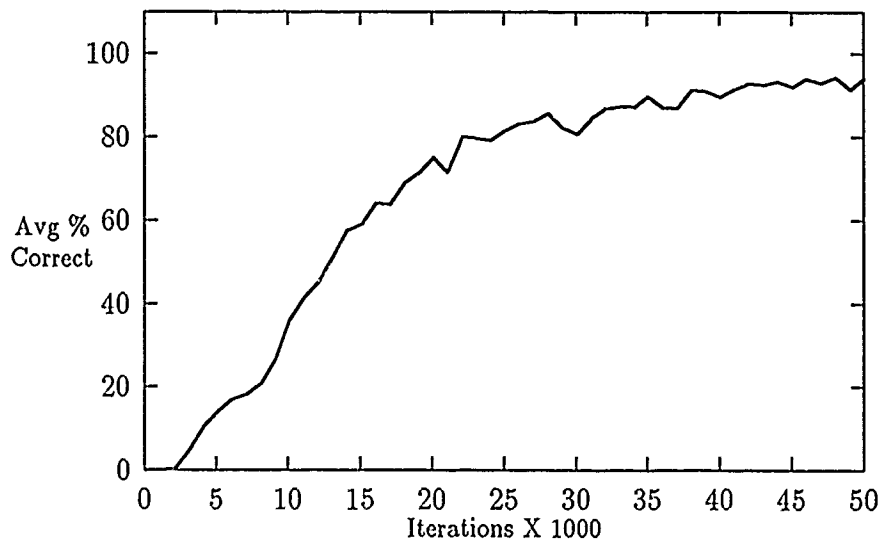

Figure 4.6. Run 4 CE Training Performance

4.5.1 Classification Accuracy on Test Vectors for Run 4. The average classification accuracy on the test vectors produced by the Center at Class Averages RBF networks and the CE back-propagation networks can be found in Table 4.9. The 12 conditional probabilities of misclassification of the test vectors and the individual networks' classification statistics can be found in Appendix A.

Table 4.9. Output Summary Statistics for Distributions of Run 4

Run ID	Statistic	$P(1 \mid 1)$	$P(2 \mid 2)$	$P(3 \mid 3)$	$P(4 \mid 4)$	P(good)
R4CA	Mean	0.7007	0.7760	0.8293	0.7720	0.7670
	STD	0.0698	0.0701	0.0498	0.0854	0.0259
R4CE	Mean	0.6260	0.7040	0.6960	0.7500	0.6940
	STD	0.0948	0.0595	0.0602	0.0474	0.0360

4.5.2 Run 4 Summary. The results of Run 4 showed that the ANNs can be trained to classify a combination of 4 classes of spread spectrum correlation signatures (DS, LSFH, RDFH, and HYB). The Center at Class Averages networks produced a classification accuracy more than 7% higher than did the CE networks. The differences in accuracy and training time combine to show that the Center at Class Averages clearly outperforms the CE networks for the classification of the Run 4 datasets. Both the Center at Class Averages and the CE networks produced a lower overall classification accuracy for the Run 4 data configuration than for the Run 3 data configuration. The addition of the class 3 and 4 vectors produced about a 5% lower classification accuracy with the Center at Class Averages networks and a more than 10% lower classification accuracy with the CE networks.

4.6 Chip Rate Results.

4.6.1 Introduction. This section contains the classification results of the 5 DS Chip Rate experiments set up in Chapter 3. The tables in this section list the chip rates of the vectors used for training in left most column with the chip rates of the test vectors listed horizontally across the top. Each table reveals the number test vectors of a given chip rate which were classified to each class or chip rate used in training.

Table 4.10. CR1 Test Vector Classification Results

Training Rates	# used for Training	Test Chip Rates(MHz)				
		1.0	1.5	2.0	2.5	3.0
1.0 MHz	16	8	2	1	0	0
1.5 MHz	12	7	9	2	2	0
2.0 MHz	06	1	1	0	0	0
2.5 MHz	11	0	0	2	4	5
3.0 MHz	06	1	0	0	5	0
Total Test Vectors		17	12	5	11	5

4.6.2 Chip Rate Classification Results. For Experiment CR1, about half of the available vectors from each of the 5 chip rates were used for training and half for testing. The overall classification accuracy was about 42% on the test vectors. Although this classification accuracy is low, the misclassified test vectors tended to be classified to a rate only 0.5 MHz different from the vectors' actual chip rate. The 1.5 MHz test vectors showed the best classification accuracy while none of the 2.0 and 3.0 MHz test vectors were correctly classified.

Table 4.11. CR2 Test Vector Classification Results

Training Rates	# used for Training	Test Chip Rates(MHz)				
		1.0	1.5	2.0	2.5	3.0
1.0 MHz	10	12	2	0	0	0
1.5 MHz	10	7	6	0	0	0
2.0 MHz	10	2	3	0	1	0
2.5 MHz	10	0	2	1	6	0
3.0 MHz	10	2	1	0	5	1
Total Test Vectors		23	14	1	12	1

The network used in Experiment CR2 produced an overall classification accuracy of 49% on the test vectors. This network, trained with 10 vectors of each class, produced results very similar to network of Experiment 1.

Table 4.12. CR3 Test Vector Classification Results

Training Rates	# used for Training	Test Chip Rates(MHz)				
		1.0	1.5	2.0	2.5	3.0
1.0 MHz	11	13	11	0	0	0
3.0 MHz	11	9	13	11	22	0
Total Test Vectors		22	24	11	22	0

The network for Experiment CR3, trained with 2 classes and tested with 5 classes, produced some unexpected results. Although the 2.5 MHz test vectors were classified as 3.0 MHz vectors as expected, the 1.0, 2.0, and 3.0 MHz test vector classifications were all skewed more than expected toward the 3.0 MHz training class.

Table 4.13. CR4 Test Vector Classification Results

Training Rates	# used for Training	Test Chip Rates(MHz)				
		1.0	1.5	2.0	2.5	3.0
1.0 MHz	18	10	13	0	0	0
2.5 MHz	18	5	11	11	4	11
Total Test Vectors		15	24	11	4	11

The network used for Experiment CR4 produced results very similar to those produced in Experiment CR3. A high proportion of the test vectors from the lower end of the 5 available chip rates were again classified as a chip rate from the upper end of the available rates.

Table 4.14. CR5 Test Vector Classification Results

Training Rates	# used for Training	Test Chip Rates(MHz)				
		1.0	1.5	2.0	2.5	3.0
1.5 MHz	18	24	1	4	1	0
2.5 MHz	18	9	5	7	3	11
Total Test Vectors		33	6	11	4	11

The Experiment CR5 network produced results in which the 1.0 and 3.0 MHz test vectors were generally classified to the nearest chip rate used for training. The 2.0 MHz test vectors classifications were split among the two nearest training class rates that were both 0.5 MHz away. The only unexpected result was that 5 out 6 of the 1.5 MHz test vectors were classified as 2.5 MHz vectors despite the fact that 1.5 MHz vectors were used in training.

4.6.3 Chip Rate Experiments Summary. The classification results of the five networks for the chip rate experiments did not approach the results achieved by the ANNs on the spread spectrum technique classifications reported earlier in this chapter. Neither five class problem achieved a classification accuracy of over 50%. For the two class problems, a pattern developed in which the test vectors from the

2.5 and 3.0 MHz classes responded as expected to training vectors of similar chip rates. However, test vectors from the lower chip rates also were often classified as 2.5 and 3.0 MHz vectors. In general, the networks produced correct classifications for some of the classes and the misclassified vectors from the five class problems tended to be classified to a chip rate as close as possible to the correct rate. These two facts suggest that further research using ANNs for classifying DS spread spectrum chip rates from correlation signatures would be useful.

4.7 Hopping Rate Experiment Results

4.7.1 Introduction. This section contains the classification results of the 4 RDFH Hopping Rate experiments set up in Chapter 3. The tables in this section list the hopping rates of the vectors used for training in left most column with the hopping rates of the test vectors listed horizontally across the top. These rates have been rounded to the nearest tenth for presentation purposes. The table for each experiment shows the number of test vectors of a given hopping rate which were classified to a class or hopping rate used in training.

4.7.2 Hopping Rate Classification Results. The network used for Experiment HR1 produced an overall classification accuracy of about 30%. The misclassified vectors were widely distributed among the incorrect classes.

The network used for Experiment HR2 show an overall classification accuracy of under 26%.

Table 4.15. HR1 Test Vector Classification Results

Training Rates (hops/sec)	# used for Training	Test Hopping Rates (hops/sec)							
		62.5	46.9	39.1	31.3	15.6	7.8	6.3	2.3
62.5	14	4	3	1	0	0	2	0	0
46.9	14	2	4	3	0	1	2	1	1
39.1	08	0	1	1	0	0	0	0	0
31.3	09	3	1	0	5	0	0	0	2
15.6	11	1	2	1	1	1	1	2	0
7.8	14	2	2	1	1	1	3	1	0
6.3	10	0	0	0	2	5	4	6	4
2.3	09	1	0	0	0	2	1	0	2
Total Test Vectors		13	13	7	9	10	13	10	9

Table 4.16. HR2 Test Vector Classification Results

Training Rates (hops/sec)	# used for Training	Test Hopping Rates (hops/sec)							
		62.5	46.9	39.1	31.3	15.6	7.8	6.3	2.3
62.5	11	4	1	0	0	2	3	1	0
46.9	11	2	1	0	0	1	0	1	1
39.1	11	3	6	3	0	1	5	0	0
31.3	11	2	0	0	5	1	3	0	1
15.6	11	1	1	0	1	2	2	3	1
7.8	11	3	3	1	0	0	2	0	0
6.3	11	1	2	0	0	0	1	3	2
2.3	11	0	2	0	1	3	0	1	2
Total Test Vectors		16	16	4	7	10	16	9	7

Table 4.17. HR3 Test Vector Classification Results

Training Rates (hops/sec)	# used for Training	Test Hopping Rates (hops/sec)							
		62.5	46.9	39.1	31.3	15.6	7.8	6.3	2.3
62.5	18	6	13	8	4	5	4	6	3
7.8	18	3	14	7	14	16	5	14	15
Total Test Vectors		9	27	15	18	21	9	20	18

The Nodes at Data Points network used for Experiment HR3 classified the test vectors from the five lowest hopping rates (2.3 hops/sec through 31.3 hops/sec) as 7.8 hops/sec vectors about 75% of the time. For the other three classes of test vectors closer to the 62.5 hops/sec training class, no pattern was observed.

Table 4.18. HR4 Test Vector Classification Results

Training Rates (hops/sec)	# used for Training	Test Hopping Rates (hops/sec)							
		62.5	46.9	39.1	31.3	15.6	7.8	6.3	2.3
46.9	24	12	3	8	3	8	0	5	2
7.8	24	15	0	7	15	13	3	15	16
Total Test Vectors		27	3	15	18	21	3	20	18

The results produced by the network used in Experiment HR4 showed that switching the higher rate training vectors from 62.5 hops/sec to 46.9 hops/sec produced virtually the same classification results from the networks. The classifications of test vectors with a hopping rate of 31.3 hops/sec were still skewed heavily toward the 7.8 hops/sec training class despite having a hopping rate closer to the 46.9 hops/sec training class.

4.7.3 Hopping Rate Experiments Summary. The classification accuracies produced by the networks trained with eight classes of hopping rates were much lower than the accuracies that the ANNs had produced for previous classification problems

in this chapter. For additional research to be warranted in the area of classifying the hopping rate of RDFH spread spectrum signals from captured correlation signatures, signatures other than the ones used in this thesis should be generated.

4.8 Conclusion

In this chapter, the results of the networks trained and tested for this thesis were presented and discussed. Chapter 5 will state the conclusions and recommendations for further research that can be drawn from the results in this chapter.

V. *Conclusions and Recommendations*

5.1 Conclusions

5.1.1 Two Class Network Performance. Radial Basis Function (RBF) neural networks can be trained directly on the correlation signatures of a selected combination of direct sequence (DS) and linearly-stepped frequency hopped (LSFH) spread spectrum signals.

Nodes at Data Points networks with a node placed at each training vector provided classification accuracies of about 90% for the test vectors as selected for Runs 1 and 2 of this thesis. Center at Class Averages networks using significantly less nodes and training time produced classification accuracies between 85% and 90% for the same data. Back-propagation networks employing the mean-squared error had achieved classification accuracies of about 80% for the same data set configurations.

5.1.2 Controlling Probability Matrix Symmetry. The conditional classification accuracies or probabilities produced by the Nodes at Data Points and the Center at Class Averages networks can be controlled by adjusting the proportions of vector classes in the training data set.

5.1.3 Data Set Selection Method Effects. Given the same vectors from which to select training and test data sets, the method used to determine which vectors will be used for training and testing the neural networks can greatly affect the classification accuracy produced by the networks.

For Run 3 of this thesis, the vectors to be used for training the individual networks were selected at random from the pool of DS and LSFH signatures. Under these conditions, the Nodes at Data Points RBF networks produced a classification accuracy of about 86%. The Center at Class Averages RBF, MSE back-propagation, and the CE back-propagation networks produced classification accuracies of about

80%. The CFM back-propagation networks produced an accuracy of only about 73%. The accuracies produced by the RBF networks of Run 3 are about 6% lower than the accuracies produced by the RBF networks of Run 1.

5.1.4 Majority Vote Results. A majority vote of the three type of back-propagation networks (MSE, CE, and CFM) trained with the exact same DS and LSFH correlation signatures did not produce a classification accuracy advantage over the individual back-propagation networks.

For Run 3 of this thesis, a majority vote network decision as stated above was produced from the classifications of the three types of back-propagation networks. The classification accuracy produced by the majority vote arbitration scheme was approximately equal to that produced by the MSE or CE networks alone.

5.1.5 Four Class Network Performance. A Center at Class Averages RBF neural network and a CE back-propagation network can be trained directly on the correlation signatures of a combination of four classes of spread spectrum signals (direct sequence (DS), linearly-stepped frequency hopped (LSFH), randomly-driven frequency hopped (RDFH) and a hybrid of DS and RDFH (HYB)). The Center at Class Averages networks produced a classification accuracy of about 77% while the CE networks produced an accuracy of near 70%.

5.1.6 Classification Accuracy. For the data configurations used in this thesis, RBF networks were trained that consistently produced overall test vector classification accuracies from 5% to 10% higher than the back-propagation networks trained and tested with the same data.

5.1.7 Training Times. For the problem of classifying spread spectrum correlation signatures with neural networks, RBF networks can be expected to train in significantly less time than back-propagation networks. The time difference will

depend on the training data sets used, the networks chosen, and the networks parameters selected.

5.1.8 Chip and Hopping Rate Networks Performance. The results were inconclusive concerning the use of ANNs for classifying the chip rate of DS and the hopping rate of RDFH spread spectrum signals from captured correlation signatures.

Although some of the chip rate and hopping rate experiment networks trained for this thesis produced high classification accuracies for some training sets containing two rates, the classification performances of the networks were generally unpredictable and unimpressive.

5.2 Recommendations

1. For future research involving the classification of spread spectrum signals using ANNs, white gaussian noise should be added to the data used to train the networks. The addition of noise would provide a more realistic test of the ANNs performance.
2. Additional research should be performed to determine if ANNs can be used to classify features of spread spectrum other than signal type. If future research is to include further attempts to classify DS chip rate or RDFH hopping rate, additional correlation signatures should be obtained.
3. An attempt should be made to develop arbitration schemes that take advantage of the differences in classifications produced by the various types of ANNs.
4. For future research, the training times of the networks should be reported due to importance of training time for many applications.

Appendix A. *Data Tables*

The following data tables were developed from the output of the various networks used in this thesis effort. The summary statistics tables of Chapter 4 are taken from the probability matrix tables in this appendix. These tables include the $P(\text{good})$ for each network as well as all conditional probabilities for each network. The training history data used to produce the training history plots presented in Chapter 4 are contained in this appendix. Finally, the data used to produce the two Center at Class Averages versus Average Threshold plots are also contained in this appendix.

Table A.1. Probability Matrices for Run 1 Center at Class Averages

Net	$P(1 \mid 1)$	$P(2 \mid 1)$	$P(1 \mid 2)$	$P(2 \mid 2)$	P(good)
net1	0.84	0.16	0.10	0.90	0.87
net2	0.74	0.26	0.12	0.88	0.81
net3	0.84	0.16	0.06	0.94	0.89
net4	0.90	0.10	0.08	0.92	0.91
net5	0.82	0.18	0.08	0.92	0.87
net6	0.86	0.14	0.08	0.92	0.89
net7	0.86	0.14	0.10	0.90	0.88
net8	0.84	0.16	0.08	0.92	0.88
net9	0.82	0.18	0.06	0.94	0.88
net10	0.82	0.18	0.10	0.90	0.86
net11	0.88	0.12	0.04	0.96	0.92
net12	0.88	0.12	0.04	0.96	0.92
net13	0.84	0.16	0.06	0.94	0.89
net14	0.88	0.12	0.08	0.92	0.90
net15	0.86	0.14	0.06	0.94	0.90
net16	0.80	0.20	0.06	0.94	0.87
net17	0.86	0.14	0.10	0.90	0.88
net18	0.84	0.16	0.04	0.96	0.90
net19	0.82	0.18	0.06	0.94	0.88
net20	0.84	0.16	0.04	0.96	0.90
net21	0.84	0.16	0.10	0.90	0.87
net22	0.82	0.18	0.08	0.92	0.87
net23	0.82	0.18	0.10	0.90	0.86
net24	0.78	0.22	0.06	0.94	0.86
net25	0.74	0.26	0.10	0.90	0.82
net26	0.82	0.18	0.08	0.92	0.87
net27	0.84	0.16	0.08	0.92	0.88
net28	0.84	0.16	0.08	0.92	0.88
net29	0.86	0.14	0.06	0.94	0.90
net30	0.84	0.16	0.06	0.94	0.89
Mean	0.8347	0.1653	0.0747	0.9253	0.8800
STD	0.0360	0.0360	0.0216	0.0216	0.0241

Table A.2. Probability Matrices for Run 1 Nodes at Data Points

Net	$P(1 \mid 1)$	$P(2 \mid 1)$	$P(1 \mid 2)$	$P(2 \mid 2)$	$P(\text{good})$
net1	0.90	0.10	0.06	0.94	0.92
net2	0.90	0.10	0.06	0.94	0.92
net3	0.90	0.10	0.06	0.94	0.92
net4	0.90	0.10	0.06	0.94	0.92
net5	0.90	0.10	0.06	0.94	0.92
Mean	0.90	0.10	0.06	0.94	0.92
STD	0.90	0.00	0.00	0.00	0.00

Table A.3. Probability Matrices for Run 2 Center at Class Averages

Net	$P(1 \mid 1)$	$P(2 \mid 1)$	$P(1 \mid 2)$	$P(2 \mid 2)$	P(good)
net1	0.90	0.10	0.10	0.90	0.90
net2	0.92	0.08	0.20	0.80	0.86
net3	0.82	0.18	0.14	0.86	0.84
net4	0.90	0.10	0.22	0.78	0.84
net5	0.84	0.16	0.12	0.88	0.86
net6	0.92	0.08	0.20	0.80	0.86
net7	0.84	0.16	0.28	0.72	0.78
net8	0.94	0.06	0.20	0.80	0.87
net9	0.90	0.10	0.12	0.88	0.89
net10	0.92	0.08	0.20	0.80	0.86
net11	0.88	0.12	0.16	0.84	0.86
net12	0.86	0.14	0.24	0.76	0.81
net13	0.84	0.16	0.20	0.80	0.82
net14	0.84	0.16	0.24	0.76	0.80
net15	0.86	0.14	0.18	0.82	0.84
net16	0.94	0.06	0.22	0.78	0.86
net17	0.84	0.16	0.18	0.82	0.83
net18	0.92	0.08	0.10	0.90	0.91
net19	0.84	0.16	0.08	0.92	0.88
net20	0.90	0.10	0.24	0.76	0.83
net21	0.90	0.10	0.12	0.88	0.89
net22	0.88	0.12	0.16	0.84	0.86
net23	0.94	0.06	0.20	0.80	0.87
net24	0.90	0.10	0.12	0.88	0.89
net25	0.90	0.10	0.16	0.84	0.87
net26	0.82	0.18	0.12	0.88	0.85
net27	0.92	0.08	0.14	0.86	0.89
net28	0.82	0.18	0.14	0.86	0.84
net29	0.88	0.12	0.18	0.82	0.85
net30	0.88	0.12	0.22	0.78	0.83
Mean	0.8820	0.1180	0.1727	0.8273	0.8547
STD	0.0384	0.0384	0.0502	0.0502	0.0300

Table A.4. Probability Matrices for Run 2 Nodes at Data Points

Net	$P(1 \mid 1)$	$P(2 \mid 1)$	$P(1 \mid 2)$	$P(2 \mid 2)$	P(good)
net1	0.90	0.10	0.10	0.90	0.90
net2	0.92	0.08	0.14	0.86	0.89
net3	0.92	0.08	0.14	0.86	0.89
net4	0.92	0.08	0.18	0.82	0.87
net5	0.92	0.08	0.10	0.90	0.91
net6	0.92	0.08	0.14	0.86	0.89
net7	0.90	0.10	0.20	0.80	0.85
net8	0.96	0.04	0.14	0.86	0.91
net9	0.92	0.08	0.10	0.90	0.91
net10	0.94	0.06	0.12	0.88	0.91
net11	0.94	0.06	0.14	0.86	0.90
net12	0.90	0.10	0.14	0.86	0.88
net13	0.94	0.06	0.18	0.82	0.88
net14	0.92	0.08	0.08	0.92	0.92
net15	0.92	0.08	0.10	0.90	0.91
net16	0.92	0.08	0.10	0.90	0.91
net17	0.90	0.10	0.12	0.88	0.89
net18	0.92	0.08	0.10	0.90	0.91
net19	0.92	0.08	0.06	0.94	0.93
net20	0.90	0.10	0.20	0.80	0.85
net21	0.94	0.06	0.08	0.92	0.93
net22	0.90	0.10	0.08	0.92	0.91
net23	0.98	0.02	0.16	0.84	0.91
net24	0.92	0.08	0.12	0.88	0.90
net25	0.96	0.04	0.14	0.86	0.91
net26	0.92	0.08	0.12	0.88	0.90
net27	0.92	0.08	0.10	0.90	0.91
net28	0.88	0.12	0.18	0.82	0.85
net29	0.92	0.08	0.08	0.92	0.92
net30	0.92	0.08	0.18	0.82	0.87
Mean	0.9220	0.0780	0.1273	0.8727	0.8973
STD	0.0206	0.0206	0.0384	0.0384	0.0220

Table A.5. Run R3MSE Training History Data

Iteration	net 1 %crct	net 2 %crct	net 3 %crct	net 4 %crct	net 5 %crct	net 6 %crct	net 7 %crct	net 8 %crct	net 9 %crct	net 10 %crct	Mean %crct	STD %crct
1000	0.00	6.86	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.69	2.17
2000	0.00	8.82	0.00	0.00	0.00	0.00	0.00	2.94	0.00	0.00	1.18	2.84
3000	0.00	28.43	0.00	1.96	31.37	0.00	0.00	7.84	0.00	12.75	8.24	12.21
4000	40.20	30.39	0.00	14.71	26.47	8.82	0.00	6.86	17.65	32.35	17.74	14.11
5000	44.18	57.84	0.00	12.75	49.02	8.82	0.00	32.35	28.43	56.86	29.02	22.62
6000	35.29	59.80	0.00	44.12	42.16	34.31	22.55	60.78	29.41	63.73	39.21	19.65
7000	71.57	67.65	0.00	63.73	74.51	65.69	22.55	70.59	38.23	81.37	55.59	26.46
8000	53.92	69.61	0.00	78.43	75.49	66.67	32.35	80.39	53.92	85.29	59.61	26.27
9000	76.47	82.35	0.00	84.31	74.51	85.29	53.92	90.20	62.75	90.20	70.00	27.24
10000	67.65	87.25	6.86	81.37	81.37	90.20	74.51	88.24	58.82	96.08	73.23	25.84
11000	57.84	92.16	13.73	79.41	86.27	92.16	92.16	92.16	79.41	99.02	78.43	25.50
12000	77.45	94.12	13.73	88.24	75.49	98.04	98.04	92.16	93.14	85.29	91.57	25.08
13000	82.35	96.08	26.47	94.12	95.10	99.02	99.02	96.08	95.10	95.10	87.84	22.06
14000	85.29	96.08	26.47	93.14	92.16	100.00	99.02	90.20	98.04	99.02	87.94	22.10
15000	92.16	97.06	15.69	93.14	97.06	100.00	100.00	96.08	99.02	100.00	89.02	25.92
16000	90.20	100.00	20.59	95.10	96.08	100.00	100.00	95.10	100.00	100.00	89.71	24.51
17000	89.22	100.00	17.65	96.07	95.12	100.00	100.00	96.08	100.00	100.00	89.41	25.46
18000	97.06	100.00	47.06	98.04	100.00	100.00	100.00	94.12	100.00	100.00	93.63	16.48
19000	92.16	100.00	55.88	99.02	100.00	100.00	100.00	96.08	100.00	100.00	94.31	13.75
20000	97.06	100.00	46.08	98.04	100.00	100.00	100.00	93.14	100.00	100.00	93.43	16.78
21000	94.12	100.00	66.67	99.02	100.00	100.00	100.00	96.08	100.00	100.00	95.59	10.37
22000	95.10	100.00	51.96	99.02	100.00	100.00	100.00	89.22	100.00	100.00	93.53	15.02
23000	92.16	100.00	80.39	98.04	100.00	100.00	100.00	96.08	100.00	100.00	96.67	6.27
24000	92.16	100.00	56.86	99.02	100.00	100.00	100.00	81.37	100.00	100.00	92.94	14.03
25000	93.14	100.00	76.47	97.06	100.00	100.00	100.00	90.20	100.00	100.00	95.69	7.60
26000	97.06	100.00	86.27	97.06	100.00	100.00	100.00	97.06	100.00	100.00	97.74	4.26
27000	100.00	100.00	89.22	99.02	100.00	100.00	100.00	99.02	100.00	100.00	98.73	3.37
28000	100.00	100.00	91.17	98.04	100.00	100.00	100.00	100.00	100.00	100.00	98.92	2.79
29000	100.00	100.00	91.17	97.06	100.00	100.00	100.00	100.00	100.00	100.00	98.82	2.84
30000	100.00	100.00	93.14	99.02	100.00	100.00	100.00	100.00	100.00	100.00	99.22	2.16
31000	100.00	100.00	93.14	96.08	100.00	100.00	100.00	100.00	100.00	100.00	98.92	2.38
32000	100.00	100.00	95.10	98.04	100.00	100.00	100.00	100.00	100.00	100.00	99.31	1.60
33000	100.00	100.00	97.06	98.04	100.00	100.00	100.00	100.00	100.00	100.00	99.51	1.06
34000	100.00	100.00	66.67	96.08	100.00	100.00	100.00	100.00	100.00	100.00	96.27	10.47
35000	100.00	100.00	96.08	97.06	100.00	100.00	100.00	100.00	100.00	100.00	99.31	1.47
36000	100.00	100.00	95.10	99.02	100.00	100.00	100.00	100.00	100.00	100.00	99.41	1.55
37000	100.00	100.00	95.10	96.08	100.00	100.00	100.00	100.00	100.00	100.00	99.12	1.87
38000	100.00	100.00	97.06	96.08	100.00	100.00	100.00	100.00	100.00	100.00	99.31	1.47
39000	100.00	100.00	53.92	94.12	100.00	100.00	100.00	100.00	100.00	100.00	94.80	14.48
40000	100.00	100.00	95.10	96.08	100.00	100.00	100.00	100.00	100.00	100.00	99.12	1.87
41000	100.00	100.00	97.06	96.08	100.00	100.00	100.00	100.00	100.00	100.00	99.31	1.47
42000	100.00	100.00	97.06	90.20	100.00	100.00	100.00	100.00	100.00	100.00	98.13	3.14
43000	100.00	100.00	80.39	96.08	100.00	100.00	100.00	100.00	100.00	100.00	97.65	6.19
44000	100.00	100.00	99.02	91.18	100.00	100.00	100.00	100.00	100.00	100.00	99.02	2.77
45000	100.00	100.00	98.04	97.06	100.00	100.00	100.00	100.00	100.00	100.00	99.51	1.06
46000	100.00	100.00	98.04	97.06	100.00	100.00	100.00	100.00	100.00	100.00	99.51	1.06
47000	100.00	100.00	96.08	74.51	100.00	100.00	100.00	100.00	100.00	100.00	97.06	8.02
48000	100.00	100.00	99.02	99.02	100.00	100.00	100.00	100.00	100.00	100.00	99.80	0.41
49000	100.00	100.00	99.02	100.00	100.00	100.00	100.00	100.00	100.00	100.00	99.90	0.31
50000	100.00	100.00	99.02	100.00	100.00	100.00	100.00	100.00	100.00	100.00	99.90	0.31

Table A.6. Run R3aMSE Training History Data

Iterations	net 1 %crct	net 2 %crct	net 3 %crct	net 4 %crct	net 5 %crct	net 6 %crct	net 7 %crct	net 8 %crct	net 9 %crct	net 10 %crct	Mean %crct	STD %crct
1000	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2000	0.00	3.92	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.39	1.24
3000	0.00	7.84	0.00	0.00	0.00	0.00	1.96	0.00	0.00	0.00	0.98	2.49
4000	3.92	6.86	0.00	3.92	2.94	0.00	2.94	0.98	0.00	0.00	2.16	2.35
5000	32.35	26.47	2.94	6.86	9.80	0.00	16.67	3.92	3.92	15.69	11.86	10.78
6000	45.10	39.22	13.73	22.55	41.18	1.96	26.47	19.61	15.69	33.33	25.88	13.79
7000	60.78	55.88	40.20	49.02	56.86	25.49	50.00	37.25	39.22	45.10	45.98	10.73
8000	69.61	66.67	58.82	70.59	67.75	42.16	56.86	52.94	46.08	61.76	59.32	9.89
9000	70.59	68.83	65.69	73.53	72.55	59.80	72.55	67.65	54.90	66.67	67.26	5.96
10000	70.59	77.45	77.45	81.37	73.53	63.73	79.41	71.57	65.69	67.65	72.84	6.02
11000	76.47	78.43	82.35	83.33	78.43	72.55	81.37	73.53	70.59	76.47	77.35	4.27
12000	87.25	83.33	76.47	82.35	86.27	78.43	86.27	78.43	78.43	82.35	81.96	3.87
13000	89.22	89.22	73.53	93.14	93.14	88.24	91.18	88.24	78.43	81.37	86.57	6.59
14000	88.24	90.20	83.33	95.10	89.22	87.25	94.12	86.27	85.29	83.33	88.23	4.06
15000	88.24	92.16	92.16	93.14	96.08	86.27	94.12	88.24	90.20	88.24	90.88	3.14
16000	90.20	93.14	94.12	95.10	98.04	94.12	96.08	88.24	94.12	89.22	93.24	3.11
17000	89.22	92.16	92.16	96.08	95.10	97.06	95.10	92.16	93.14	91.18	93.34	2.44
18000	92.16	93.14	96.08	97.06	99.02	98.04	98.04	89.22	90.20	98.04	95.10	3.61
19000	88.24	95.10	96.08	96.08	100.00	100.00	98.04	93.14	96.08	97.06	95.98	3.44
20000	91.18	93.14	98.04	99.20	100.00	100.00	98.04	90.20	99.20	100.00	96.90	3.86
21000	85.29	93.14	98.04	97.06	100.00	100.00	97.06	96.08	99.20	100.00	96.59	4.52
22000	91.18	93.14	98.04	96.08	100.00	100.00	99.02	92.16	99.02	100.00	96.86	3.48
23000	86.27	91.18	99.02	98.04	100.00	100.00	100.00	97.06	100.00	100.00	97.16	4.70
24000	90.20	92.16	99.02	98.04	100.00	100.00	100.00	93.14	100.00	100.00	97.26	3.86
25000	89.22	78.43	100.00	98.04	100.00	100.00	100.00	97.06	100.00	100.00	96.27	7.11
26000	94.12	87.25	100.00	98.04	100.00	100.00	100.00	98.04	100.00	100.00	97.74	4.14
27000	93.14	93.14	100.00	98.04	100.00	100.00	100.00	94.12	100.00	100.00	97.84	3.09
28000	94.12	93.14	100.00	98.04	100.00	100.00	100.00	97.06	100.00	100.00	98.24	2.65
29000	94.12	84.31	100.00	87.25	100.00	100.00	100.00	97.06	100.00	100.00	96.27	5.90
30000	96.08	90.20	100.00	96.08	100.00	100.00	100.00	98.04	100.00	100.00	98.04	3.20
31000	96.08	82.35	100.00	95.10	100.00	100.00	100.00	95.10	100.00	100.00	96.86	5.54
32000	96.08	90.20	100.00	96.08	100.00	100.00	100.00	97.06	100.00	100.00	97.94	3.22
33000	96.08	89.22	100.00	95.10	100.00	100.00	100.00	99.02	100.00	100.00	97.94	3.56
34000	93.14	94.12	100.00	94.12	100.00	100.00	100.00	96.08	100.00	100.00	97.75	3.00
35000	95.10	92.16	100.00	87.25	100.00	100.00	100.00	94.12	100.00	100.00	96.86	4.52
36000	93.14	86.27	100.00	93.14	100.00	100.00	100.00	98.04	100.00	100.00	97.06	4.71
37000	91.18	90.20	100.00	89.22	100.00	100.00	100.00	99.02	100.00	100.00	96.96	4.70
38000	94.12	89.22	100.00	91.18	100.00	100.00	100.00	98.04	100.00	100.00	97.26	4.18
39000	94.12	95.10	100.00	88.24	100.00	100.00	100.00	99.02	100.00	100.00	97.65	3.98
40000	93.14	90.20	100.00	86.27	100.00	100.00	100.00	95.10	100.00	100.00	96.47	5.07
41000	91.18	95.10	100.00	78.43	100.00	100.00	100.00	95.10	100.00	100.00	95.98	6.91
42000	87.25	98.04	100.00	85.29	100.00	100.00	100.00	93.14	100.00	100.00	96.37	5.76
43000	71.57	98.04	100.00	91.18	100.00	100.00	100.00	90.20	100.00	100.00	95.10	9.10
44000	86.27	99.02	100.00	96.08	100.00	100.00	100.00	84.31	100.00	100.00	96.57	6.09
45000	93.14	99.02	100.00	90.20	100.00	100.00	100.00	93.14	100.00	100.00	97.55	3.82
46000	91.18	99.02	100.00	90.20	100.00	100.00	100.00	98.04	100.00	100.00	97.84	3.83
47000	94.12	100.00	100.00	96.08	100.00	100.00	100.00	97.06	100.00	100.00	98.73	2.17
48000	96.08	100.00	100.00	95.10	100.00	100.00	100.00	100.00	100.00	100.00	99.12	1.87
49000	97.06	100.00	100.00	97.06	100.00	100.00	100.00	99.02	100.00	100.00	99.31	1.23
50000	97.06	100.00	100.00	96.08	100.00	100.00	100.00	100.00	100.00	100.00	99.31	1.47

Table A.7. Run R3aCE Training History Data

Iteration	net 1 %crct	net 2 %crct	net 3 %crct	net 4 %crct	net 5 %crct	net 6 %crct	net 7 %crc	net 8 %crct	net 9 %crct	net 10 %crct	Mean %crct	STD %crct
1000	0.00	6.86	9.00	2.94	0.00	0.00	5.88	8.82	0.00	0.00	2.45	3.29
2000	19.61	25.49	31.37	20.59	18.63	16.67	27.45	11.76	40.20	9.80	22.16	8.73
3000	52.94	54.90	63.73	44.12	76.47	39.22	54.90	52.94	47.06	55.88	54.22	9.85
4000	95.10	73.53	46.08	83.33	90.20	40.20	78.43	74.51	60.78	69.61	71.18	16.89
5000	95.10	91.18	57.84	96.08	91.18	57.84	89.22	80.39	75.49	81.37	81.57	13.42
6000	95.10	84.31	91.18	90.20	97.06	67.65	90.20	79.41	89.22	95.10	87.94	8.40
7000	99.02	88.24	91.18	92.16	96.08	79.41	92.16	100.00	84.31	100.00	92.25	6.52
8000	99.02	97.06	97.06	89.22	93.14	79.41	99.02	100.00	93.14	100.00	94.71	6.11
9000	98.04	96.08	100.00	97.06	100.00	94.12	100.00	100.00	96.08	100.00	98.14	2.08
10000	99.02	99.02	100.00	96.08	100.00	99.02	100.00	100.00	83.33	100.00	97.65	4.91
11000	100.00	100.00	100.00	98.04	100.00	100.00	100.00	100.00	96.08	100.00	99.41	1.26
12000	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	78.41	100.00	97.84	6.48
13000	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	92.16	100.00	99.22	2.35
14000	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	91.18	100.00	99.12	2.65
15000	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	98.04	100.00	99.80	0.59
16000	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	95.10	100.00	99.51	1.47
17000	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	92.16	100.00	99.22	2.35
18000	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	95.10	100.00	99.51	1.47
19000	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	98.04	100.00	99.80	0.59
20000	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	97.06	100.00	99.71	0.88
21000	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	99.02	100.00	99.90	0.29
22000	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	94.12	100.00	99.41	1.76
23000	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	94.12	100.00	99.41	1.76
24000	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	97.06	100.00	99.71	0.88
25000	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	99.02	100.00	99.90	0.29
26000	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	99.02	100.00	99.90	0.29
27000	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	99.02	100.00	99.90	0.29
28000	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	0.00
29000	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	0.00
30000	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	0.00

Table A.8. Run R3aCFM Training History Data

Iteration	net 1 %crct	net 2 %crct	net 3 %crct	net 4 %crct	net 5 %crct	net 6 %crct	net 7 %crct	net 8 %crct	net 9 %crct	net 10 %crct	Mean %crct	STD %crct
1000	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2000	6.86	37.25	0.00	38.24	33.33	0.00	12.75	15.69	0.00	0.00	14.41	15.30
3000	14.71	45.10	19.61	44.12	43.14	0.00	40.20	16.67	1.96	16.67	24.22	16.59
4000	11.76	49.02	17.65	49.02	47.06	21.57	37.25	34.31	17.65	6.86	29.22	15.24
5000	31.37	50.98	33.33	52.94	48.04	41.18	36.27	36.27	31.37	28.43	39.02	8.36
6000	54.90	50.98	43.14	53.92	49.02	48.04	44.12	37.25	37.25	37.25	45.59	6.48
7000	61.76	48.04	43.14	50.00	51.96	49.02	39.22	39.22	49.02	38.24	46.96	6.88
8000	68.63	46.08	49.02	36.27	52.94	50.98	44.12	48.04	62.75	46.08	50.49	8.81
9000	74.51	29.41	61.76	46.08	44.12	50.00	45.10	54.91	66.67	46.08	51.86	12.33
10000	78.43	36.27	71.57	41.18	54.90	53.92	50.98	55.88	74.51	50.00	56.76	13.26
11000	79.41	52.94	81.37	52.94	58.82	55.88	60.78	52.94	77.45	56.86	62.94	11.09
12000	84.31	59.80	77.45	52.94	58.82	54.90	65.69	53.92	78.43	58.82	64.51	10.86
13000	83.33	62.75	78.43	54.90	63.72	55.88	68.63	62.75	81.37	57.84	66.96	10.04
14000	83.33	63.73	83.33	50.00	65.69	55.88	71.59	70.59	82.35	54.90	68.14	11.67
15000	85.29	64.71	87.25	50.98	64.71	56.86	76.47	76.47	87.25	60.78	71.08	12.56
16000	83.33	71.57	88.24	58.82	67.65	56.86	76.47	79.41	85.29	64.71	73.24	10.51
17000	84.31	70.59	89.22	61.76	69.61	57.84	80.39	83.33	87.25	64.71	74.90	10.78
18000	85.29	76.47	89.22	65.69	67.65	57.84	80.39	83.33	87.25	65.69	75.88	10.36
19000	85.29	90.20	90.20	72.55	71.57	56.86	86.27	82.35	88.24	63.73	78.73	11.22
20000	86.27	79.41	91.18	74.51	71.57	57.84	87.25	83.33	88.24	76.47	79.61	9.48
21000	87.25	81.37	91.18	75.49	72.55	58.82	87.25	86.27	89.22	77.45	80.69	9.39
22000	87.25	83.33	91.18	79.41	72.55	58.82	88.24	89.22	89.22	80.39	81.96	9.45
23000	87.25	84.31	90.02	81.37	74.51	58.82	88.24	90.20	89.22	85.29	82.92	9.23
24000	87.25	86.27	91.18	87.25	73.53	49.02	88.24	91.18	89.22	81.37	82.45	12.21
25000	87.25	85.29	91.18	90.20	73.53	53.92	88.24	91.18	89.22	85.29	83.53	11.01
26000	87.25	85.29	91.18	89.22	74.51	52.94	88.24	91.18	89.22	90.20	83.92	11.32
27000	87.25	86.27	91.18	89.22	74.51	54.90	88.24	91.18	89.22	90.20	84.22	10.80
28000	87.25	83.33	97.06	91.18	74.51	51.96	88.24	91.18	89.22	91.17	84.51	12.23
29000	87.25	83.33	91.18	89.22	74.51	52.94	88.24	91.18	89.22	90.20	83.73	11.31
30000	87.25	86.27	91.18	92.16	74.51	55.88	88.24	91.18	89.22	93.14	84.90	10.89
31000	87.25	86.27	91.18	92.16	74.51	64.71	88.24	91.18	89.22	93.14	85.79	8.63
32000	87.25	87.25	91.18	93.14	74.51	68.63	88.24	91.18	89.22	93.14	86.37	7.79
33000	87.25	87.25	91.18	92.16	74.51	72.55	88.24	91.18	89.22	92.16	86.57	6.77
34000	87.25	87.25	91.18	92.16	74.51	73.53	88.24	91.18	89.22	93.14	86.77	6.65
35000	87.25	89.22	91.18	94.12	74.51	78.43	88.24	91.18	89.22	92.16	86.57	6.77
36000	87.25	90.20	91.18	92.16	74.51	81.37	88.24	91.18	89.22	93.14	87.85	5.45
37000	87.25	89.22	91.18	93.14	74.51	82.35	88.24	91.18	89.22	93.14	87.94	5.39
38000	87.25	90.20	91.18	93.14	74.51	83.33	88.24	91.18	89.22	93.14	88.14	5.33
39000	87.25	90.20	89.22	94.12	74.51	83.33	88.24	91.18	89.22	93.14	88.04	5.35
40000	87.25	91.18	90.20	94.12	74.51	79.41	88.24	91.18	89.22	93.14	87.85	5.89
41000	87.25	90.20	91.18	94.12	74.51	85.29	88.24	91.18	89.22	93.14	88.43	5.28
42000	87.25	91.18	91.18	94.12	74.51	85.29	88.24	91.18	89.22	93.14	88.53	5.32
43000	87.25	90.20	91.18	94.12	74.51	85.29	88.24	91.18	89.22	93.14	88.43	5.28
44000	87.25	91.18	91.18	94.12	74.51	86.27	88.24	91.18	89.22	93.14	88.63	5.27
45000	87.25	92.16	91.18	94.12	74.51	86.27	88.24	91.18	89.22	93.14	88.73	5.32
46000	87.25	92.16	91.18	94.12	74.51	86.27	88.24	91.18	89.22	93.14	88.73	5.32
47000	87.25	92.16	92.16	94.12	74.51	86.27	88.24	91.18	89.22	93.14	88.83	5.37
48000	87.25	92.16	92.16	94.12	74.51	86.27	88.24	91.18	89.22	93.14	88.83	5.37
49000	87.25	92.16	92.16	94.12	74.51	86.27	88.24	91.18	89.22	93.14	88.83	5.37
50000	87.25	92.16	92.16	94.12	74.51	86.27	88.24	91.18	89.22	93.14	88.83	5.37

Table A.9. Run 3 Center at Class Averages - Seed 6

Average Threshold	Number of Nodes	P(good) Training	P(good) Test
0.50	90	1.0000	0.8900
1.00	75	1.0000	0.8800
1.50	54	0.9902	0.8500
2.00	32	0.9314	0.8500
2.50	20	0.9412	0.8500
3.00	12	0.8333	0.8200
3.50	8	0.6961	0.7100
4.00	3	0.5000	0.5000

Table A.10. Probability Matrices for Run 3 Center at Class Averages

Net	$P(1 \mid 1)$	$P(2 \mid 1)$	$P(1 \mid 2)$	$P(2 \mid 2)$	P(good)
net1	0.82	0.18	0.26	0.74	0.78
net2	0.90	0.10	0.24	0.76	0.83
net3	0.74	0.26	0.20	0.80	0.77
net4	0.80	0.20	0.30	0.70	0.75
net5	0.86	0.14	0.24	0.76	0.81
net6	0.90	0.10	0.20	0.80	0.85
net7	0.92	0.08	0.26	0.74	0.83
net8	0.84	0.16	0.28	0.72	0.78
net9	0.80	0.20	0.20	0.80	0.80
net10	1.00	0.00	0.28	0.72	0.86
net11	0.94	0.06	0.28	0.72	0.83
net12	0.90	0.10	0.26	0.74	0.82
net13	0.94	0.06	0.24	0.76	0.85
net14	0.80	0.20	0.16	0.84	0.82
net15	0.94	0.06	0.30	0.70	0.82
net16	0.80	0.20	0.18	0.82	0.81
net17	0.92	0.08	0.14	0.86	0.89
net18	0.90	0.10	0.22	0.78	0.84
net19	0.92	0.08	0.28	0.72	0.82
net20	0.88	0.12	0.18	0.82	0.85
net21	0.74	0.26	0.34	0.66	0.70
net22	0.88	0.12	0.20	0.80	0.84
net23	0.92	0.08	0.22	0.78	0.85
net24	0.80	0.20	0.18	0.82	0.81
net25	0.92	0.08	0.28	0.72	0.82
net26	0.92	0.08	0.26	0.74	0.83
net27	0.98	0.02	0.26	0.74	0.86
net28	0.98	0.02	0.50	0.50	0.74
net29	0.86	0.14	0.14	0.86	0.86
net30	0.86	0.14	0.22	0.78	0.82
Mean	0.8793	0.1207	0.2433	0.7567	0.818
STD	0.0680	0.0680	0.0693	0.0693	0.0400

Table A.11. Probability Matrices for Run 3 Nodes at Data Points

Net	$P(1 \mid 1)$	$P(2 \mid 1)$	$P(1 \mid 2)$	$P(2 \mid 2)$	P(good)
net1	0.86	0.14	0.24	0.76	0.81
net2	0.78	0.22	0.24	0.76	0.77
net3	0.86	0.14	0.12	0.88	0.87
net4	0.78	0.22	0.12	0.88	0.83
net5	would	not	train		
net6	0.94	0.06	0.16	0.84	0.89
net7	0.90	0.10	0.20	0.80	0.85
net8	0.86	0.14	0.14	0.86	0.86
net9	0.88	0.12	0.20	0.80	0.84
net10	0.96	0.04	0.12	0.88	0.92
net11	would	not	train		
net12	0.84	0.16	0.12	0.88	0.86
net13	0.94	0.06	0.14	0.86	0.90
net14	0.88	0.12	0.10	0.90	0.89
net15	0.90	0.10	0.20	0.80	0.85
net16	0.94	0.06	0.14	0.86	0.90
net17	0.92	0.08	0.14	0.86	0.89
net18	0.90	0.10	0.08	0.92	0.91
net19	0.92	0.08	0.22	0.78	0.85
net20	0.86	0.14	0.14	0.86	0.86
net21	would	not	train		
net22	0.92	0.08	0.12	0.88	0.90
net23	0.90	0.10	0.18	0.82	0.86
net24	0.82	0.18	0.12	0.88	0.85
net25	0.96	0.04	0.12	0.88	0.92
net26	0.68	0.32	0.14	0.86	0.77
net27	0.94	0.06	0.10	0.90	0.92
net28	0.92	0.08	0.32	0.68	0.80
net29	0.86	0.14	0.14	0.86	0.86
net30	0.86	0.14	0.10	0.90	0.88
Mean	0.8807	0.1193	0.1541	0.8459	0.8633
STD	0.0629	0.0629	0.0546	0.0546	0.0416

Table A.12. Probability Matrices for R3MSE Networks

Net	$P(1 \mid 1)$	$P(2 \mid 1)$	$P(1 \mid 2)$	$P(2 \mid 2)$	$P(\text{good})$
net1	0.72	0.28	0.30	0.70	0.71
net2	0.64	0.36	0.20	0.80	0.72
net3	0.70	0.30	0.08	0.92	0.81
net4	0.72	0.28	0.12	0.88	0.80
net5	0.64	0.36	0.16	0.84	0.74
net6	0.90	0.10	0.22	0.78	0.84
net7	0.78	0.22	0.14	0.86	0.82
net8	0.76	0.24	0.14	0.86	0.81
net9	0.84	0.16	0.20	0.80	0.82
net10	0.86	0.14	0.24	0.76	0.81
Mean	0.7560	0.2440	0.1800	0.8200	0.7880
STD	0.0893	0.0893	0.0646	0.0646	0.0464

Table A.13. Probability Matrices for R3aMSE Networks

Net	$P(1 \mid 1)$	$P(2 \mid 1)$	$P(1 \mid 2)$	$P(2 \mid 2)$	P(good)
net1	0.78	0.22	0.24	0.76	0.77
net2	0.64	0.36	0.22	0.78	0.71
net3	0.86	0.14	0.12	0.88	0.87
net4	0.74	0.26	0.14	0.86	0.80
net5	0.78	0.22	0.16	0.84	0.81
net6	0.90	0.10	0.16	0.84	0.87
net7	0.80	0.20	0.20	0.80	0.80
net8	0.78	0.22	0.14	0.86	0.82
net9	0.90	0.10	0.16	0.84	0.87
net10	0.94	0.06	0.22	0.78	0.86
Mean	0.8120	0.1880	0.1760	0.8240	0.8180
STD	0.0895	0.0895	0.0409	0.0409	0.0522

Table A.14. Probability Matrices for R3aCE Networks

Net	$P(1 \mid 1)$	$P(2 \mid 1)$	$P(1 \mid 2)$	$P(2 \mid 2)$	P(good)
net1	0.82	0.18	0.28	0.72	0.77
net2	0.66	0.34	0.24	0.76	0.71
net3	0.82	0.18	0.12	0.88	0.85
net4	0.76	0.24	0.16	0.84	0.80
net5	0.84	0.16	0.18	0.82	0.83
net6	0.94	0.06	0.18	0.82	0.88
net7	0.84	0.16	0.18	0.82	0.83
net8	0.72	0.28	0.16	0.84	0.78
net9	0.92	0.08	0.22	0.78	0.85
net10	0.88	0.12	0.24	0.76	0.82
Mean	0.8200	0.1800	0.1960	0.8040	0.8120
STD	0.0869	0.0869	0.0479	0.0479	0.0489

Table A.15. Probability Matrices for R3aCFM Networks

Net	$P(1 \mid 1)$	$P(2 \mid 1)$	$P(1 \mid 2)$	$P(2 \mid 2)$	P(good)
net1	0.48	0.52	0.34	0.66	0.57
net2	0.78	0.22	0.30	0.70	0.74
net3	0.70	0.30	0.16	0.84	0.77
net4	0.66	0.34	0.18	0.82	0.74
net5	0.88	0.12	0.64	0.36	0.62
net6	0.88	0.12	0.44	0.56	0.72
net7	0.72	0.28	0.30	0.70	0.71
net8	0.92	0.08	0.28	0.72	0.82
net9	0.76	0.24	0.16	0.84	0.80
net10	0.94	0.06	0.28	0.72	0.83
Mean	0.7720	0.2280	0.3080	0.6920	0.7320
STD	0.1412	0.1412	0.1455	0.1455	0.0836

Table A.16. Probability Matrices for Run 3 Majority Vote

Net	$P(1 \mid 1)$	$P(2 \mid 1)$	$P(1 \mid 2)$	$P(2 \mid 2)$	P(good)
net1	0.78	0.22	0.28	0.72	0.75
net2	0.68	0.32	0.24	0.76	0.72
net3	0.86	0.14	0.12	0.88	0.87
net4	0.72	0.28	0.16	0.84	0.78
net5	0.86	0.14	0.20	0.80	0.83
net6	0.92	0.08	0.22	0.78	0.85
net7	0.76	0.24	0.20	0.80	0.78
net8	0.82	0.18	0.16	0.84	0.83
net9	0.88	0.12	0.16	0.84	0.86
net10	0.92	0.08	0.24	0.76	0.84
Mean	0.8200	0.1800	0.1980	0.8020	0.8110
STD	0.0827	0.0827	0.0485	0.0485	0.0504

Table A.17. Run 4 Center at Class Averages - Seed 1

Average Threshold	Number of Nodes	P(good) Training	P(good) Test
0.25	196	1.0000	0.7900
0.50	178	1.0000	0.7900
0.75	166	1.0000	0.7950
1.00	154	1.0000	0.7950
1.25	131	1.0000	0.7950
1.50	112	0.9804	0.7600
1.75	84	0.8725	0.6600
2.00	64	0.7745	0.6250

Table A.18. Run 4 CE Training History Data

Iteration	net 1 %cret	net 2 %cret	net 3 %cret	net 4 %cret	net 5 %cret	net 6 %cret	net 7 %cret	net 8 %cret	net 9 %cret	net10 %cret	Mean %cret	STD %cret
1000	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2000	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	2.45	0.25	0.77
3000	0.49	0.00	8.33	16.67	0.00	0.00	2.45	9.80	0.00	8.82	4.66	5.87
4000	11.27	12.75	8.82	14.22	0.00	9.31	9.31	12.25	9.80	16.67	10.44	4.44
5000	15.20	15.69	10.78	18.63	5.39	13.24	13.73	15.19	16.67	14.22	13.87	3.64
6000	17.65	18.14	19.61	20.59	12.25	13.24	13.73	15 0	19.61	17.65	16.77	2.96
7000	16.67	18.14	27.45	18.14	18.63	13.73	14.22	17.65	19.61	16.18	18.04	3.80
8000	19.12	22.06	17.16	22.06	21.08	13.73	16.67	22.55	27.94	23.53	20.59	4.05
9000	20.59	18.63	24.51	31.60	19.61	15.69	37.75	29.41	32.84	32.84	26.67	7.81
10000	31.36	34.80	25.49	38.24	36.27	19.61	39.22	40.20	51.47	41.18	35.83	8.80
11000	39.71	25.98	40.20	42.16	50.00	29.90	40.69	37.75	58.33	49.02	41.37	9.47
12000	49.51	31.86	29.41	44.61	54.41	34.31	50.49	49.51	54.90	50.49	44.95	9.53
13000	53.92	35.29	37.75	43.63	66.18	44.11	63.24	53.92	63.73	49.51	51.13	10.98
14000	64.22	53.43	53.92	47.55	58.33	50.49	68.14	57.35	64.71	55.88	57.40	6.59
15000	68.63	50.98	45.59	51.96	73.53	50.00	73.53	53.45	68.14	49.51	58.83	11.01
16000	67.65	50.49	70.59	63.24	79.90	50.43	69.12	62.75	73.64	52.94	64.01	10.05
17000	67.16	53.43	61.27	69.61	57.84	66.18	76.96	38.73	78.43	67.65	63.63	11.69
18000	66.18	56.37	77.45	77.45	72.55	54.41	79.41	58.33	71.57	74.02	68.77	9.37
19000	75.49	56.86	76.96	77.45	66.67	57.84	87.25	72.06	77.94	63.24	71.18	9.79
20000	69.18	61.27	77.94	85.29	73.53	66.18	87.75	68.63	80.88	78.43	74.91	8.60
21000	79.41	42.65	82.84	83.33	81.86	66.18	86.76	61.76	58.82	70.10	71.37	14.10
22000	81.37	62.25	76.47	84.80	86.76	75.00	90.69	77.45	83.33	80.39	79.85	7.85
23000	80.88	65.20	76.96	87.25	83.82	74.02	89.22	75.49	78.92	83.82	79.56	7.08
24000	85.29	65.20	69.61	86.27	88.73	72.06	88.24	82.35	77.45	76.47	79.17	8.29
25000	86.76	67.65	78.92	93.14	86.27	72.06	93.63	80.39	85.29	69.12	81.32	9.36
26000	85.29	69.12	80.88	90.20	89.22	73.53	93.14	84.31	83.33	81.37	83.04	7.38
27000	88.24	67 16	84.80	91.67	87.75	68.14	94.61	82.84	88.24	82.84	83.63	9.18
28000	83.33	74.51	81.37	95.59	92.16	76.96	95.59	83.33	88.24	84.31	85.54	7.28
29000	49.51	69.61	77.45	95.10	91.18	80.39	94.61	88.24	88.24	86.76	82.11	13.94
30000	53.43	80.88	73.53	96.57	85.78	78.43	91.67	77.45	82.35	86.27	80.64	11.75
31000	58.33	81.86	85.29	94.12	91.18	75.98	94.61	83.82	89.71	89.71	84.46	10.85
32000	68.63	85.78	84.80	95.59	97.06	79.90	93.63	85.78	91.18	84.31	86.67	8.42
33000	78.92	78.92	82.84	95.59	97.55	81.86	94.61	81.37	90.20	89.71	87.16	7.20
34000	78.92	78.92	83.33	98.04	94.61	75.49	95.10	84.80	90.20	90.69	87.01	7.82
35000	88.24	81.37	86.76	98.53	94.12	80.88	96.57	86.27	92.65	91.67	89.71	6.03
36000	88.73	65.69	87.25	98.04	93.14	76.47	96.08	85.78	90.20	87.75	86.91	9.56
37000	90.20	73.04	72.06	98.53	94.12	82.35	95.59	75.98	94.61	92.65	86.91	10.10
38000	94.61	90.69	85.29	96.08	97.06	83.33	96.08	84.31	94.18	90.69	91.23	5.24
39000	95.59	90.69	79.90	98.04	98.04	82.84	94.61	86.76	93.63	88.24	90.83	6.28
40000	95.59	89.71	87.75	97.06	85.78	80.88	95.59	84.80	94.61	83.82	89.56	5.80
41000	95.59	88.73	89.22	96.08	96.08	79.41	95.59	87.25	94.18	91.18	91.33	5.37
42000	94.61	94.61	89.71	98.04	94.61	83.33	96.57	88.73	95.10	91.18	92.75	4.14
43000	94.12	91.67	92.65	95.14	97.06	83.33	97.06	89.71	95.59	89.71	92.40	4.14
44000	95.59	94.12	94.12	96.57	97.55	83.82	96.57	88.73	95.59	89.71	93.24	4.41
45000	96.08	85.29	93.14	96.08	97.55	82.35	94.61	89.22	93.63	91.18	91.91	4.96
46000	97.06	95.59	90.69	98.53	98.04	86.27	96.57	90.20	95.10	90.69	93.87	4.12
47000	96.08	91.18	94.12	94.61	99.02	85.29	96.08	86.27	94.61	91.67	92.89	4.36
48000	97.06	98.53	92.16	97.06	99.02	84.80	97.06	89.71	94.61	92.65	94.27	4.49
49000	96.57	98.53	93.14	98.04	98.53	85.78	97.06	88.73	96.08	60.29	91.27	11.72
50000	98.04	96.08	93.14	98.53	99.51	86.27	93 14	89 71	97 55	88 24	94 02	4 68

Table A.19. Probabilities for Run 4 Center at Class Averages

Net	P(1 | 1)	P(2 | 1)	P(3 | 1)	P(4 | 1)	P(1 | 2)	P(2 | 2)	P(3 | 2)	P(4 | 2)
net1	0.72	0.10	0.12	0.06	0.06	0.86	0.08	0.00
net2	0.72	0.10	0.12	0.06	0.06	0.86	0.08	0.00
net3	0.82	0.00	0.06	0.12	0.24	0.70	0.04	0.02
net4	0.68	0.20	0.02	0.10	0.14	0.80	0.06	0.00
net5	0.82	0.10	0.06	0.02	0.26	0.68	0.06	0.00
net6	0.70	0.14	0.14	0.02	0.16	0.72	0.12	0.00
net7	0.78	0.02	0.06	0.14	0.28	0.68	0.04	0.00
net8	0.68	0.10	0.16	0.06	0.08	0.80	0.08	0.04
net9	0.58	0.14	0.14	0.14	0.14	0.70	0.12	0.04
net10	0.63	0.14	0.16	0.08	0.12	0.82	0.06	0.00
net11	0.74	0.06	0.08	0.12	0.16	0.68	0.14	0.02
net12	0.58	0.22	0.14	0.06	0.08	0.86	0.06	0.00
net13	0.72	0.06	0.00	0.22	0.12	0.64	0.24	0.00
net14	0.78	0.08	0.06	0.08	0.22	0.68	0.10	0.00
net15	0.64	0.18	0.06	0.12	0.20	0.74	0.02	0.04
net16	0.76	0.06	0.06	0.12	0.14	0.72	0.06	0.08
net17	0.72	0.08	0.18	0.02	0.16	0.74	0.10	0.00
net18	0.74	0.06	0.12	0.08	0.12	0.84	0.04	0.00
net19	0.64	0.10	0.12	0.14	0.16	0.76	0.08	0.00
net20	0.76	0.12	0.06	0.06	0.18	0.74	0.08	0.00
net21	0.6	0.04	0.20	0.10	0.20	0.76	0.04	0.00
net22	0.72	0.12	0.10	0.06	0.10	0.78	0.12	0.00
net23	0.76	0.04	0.10	0.10	0.26	0.66	0.06	0.02
net24	0.56	0.08	0.04	0.32	0.12	0.82	0.06	0.00
net25	0.62	0.02	0.04	0.32	0.14	0.82	0.02	0.02
net26	0.60	0.14	0.06	0.20	0.02	0.88	0.08	0.02
net27	0.74	0.10	0.06	0.10	0.12	0.84	0.02	0.02
net28	0.72	0.04	0.14	0.10	0.12	0.74	0.10	0.04
net29	0.72	0.04	0.08	0.16	0.14	0.84	0.00	0.02
net30	0.72	0.12	0.02	0.14	0.10	0.82	0.02	0.06
Mean	0.7007	0.0933	0.0920	0.1140	0.1467	0.7660	0.0727	0.0147
STD	0.0698	0.0531	0.0508	0.0736	0.0627	0.0701	0.0468	0.0210

Table A.20. Probabilities for Run 4 Center at Class Averages

Net	P(1 | 3)	P(2 | 3)	P(3 | 3)	P(4 | 3)	P(1 | 4)	P(2 | 4)	P(3 | 4)	P(4 | 4)	P(good)
net1	0.02	0.06	0.88	0.04	0.24	0.00	0.04	0.72	0.795
net2	0.02	0.06	0.88	0.04	0.24	0.00	0.04	0.72	0.795
net3	0.06	0.06	0.82	0.06	0.16	0.00	0.00	0.84	0.795
net4	0.08	0.10	0.82	0.00	0.12	0.00	0.02	0.86	0.790
net5	0.06	0.12	0.82	0.00	0.18	0.02	0.06	0.74	0.765
net6	0.00	0.10	0.88	0.02	0.08	0.00	0.06	0.86	0.790
net7	0.08	0.02	0.86	0.04	0.12	0.02	0.10	0.76	0.770
net8	0.04	0.08	0.86	0.02	0.08	0.04	0.16	0.72	0.765
net9	0.06	0.12	0.74	0.08	0.06	0.00	0.04	0.90	0.730
net10	0.08	0.12	0.76	0.04	0.08	0.06	0.02	0.84	0.760
net11	0.08	0.12	0.76	0.04	0.16	0.02	0.14	0.68	0.715
net12	0.02	0.18	0.80	0.00	0.14	0.06	0.14	0.66	0.725
net13	0.04	0.04	0.86	0.06	0.10	0.00	0.04	0.86	0.770
net14	0.06	0.04	0.90	0.00	0.12	0.02	0.10	0.76	0.780
net15	0.02	0.04	0.88	0.06	0.16	0.04	0.06	0.74	0.750
net16	0.02	0.04	0.84	0.10	0.08	0.02	0.10	0.80	0.780
net17	0.10	0.04	0.82	0.04	0.18	0.04	0.22	0.56	0.710
net18	0.02	0.18	0.80	0.00	0.24	0.00	0.08	0.68	0.765
net19	0.06	0.10	0.82	0.02	0.12	0.02	0.12	0.74	0.740
net20	0.06	0.02	0.88	0.04	0.06	0.00	0.10	0.84	0.805
net21	0.06	0.04	0.82	0.08	0.10	0.00	0.10	0.80	0.760
net22	0.02	0.18	0.74	0.06	0.14	0.00	0.06	0.80	0.760
net23	0.06	0.08	0.84	0.02	0.08	0.00	0.02	0.90	0.790
net24	0.02	0.12	0.84	0.02	0.04	0.02	0.06	0.88	0.775
net25	0.10	0.04	0.84	0.02	0.08	0.02	0.06	0.84	0.780
net26	0.06	0.04	0.90	0.00	0.08	0.16	0.04	0.72	0.775
net27	0.06	0.10	0.82	0.00	0.32	0.02	0.00	0.66	0.765
net28	0.06	0.06	0.86	0.02	0.12	0.02	0.04	0.82	0.785
net29	0.06	0.12	0.70	0.12	0.18	0.04	0.14	0.64	0.725
net30	0.02	0.14	0.84	0.00	0.12	0 00	0 06	0 82	0 800
Mean	0.0507	0.0853	0.8293	0.0347	0.1327	0.0213	0.0740	0.7720	0.7670
STD	0.0272	0.0475	0.0498	0.0319	0.0644	0.0319	0.0510	0.0854	0.0259

Table A.21. Probabilities for Run 4 Cross Entropy

Net	$P(1 \mid 1)$	$P(2 \mid 1)$	$P(3 \mid 1)$	$P(4 \mid 1)$	$P(1 \mid 2)$	$P(2 \mid 2)$	$P(3 \mid 2)$	$P(4 \mid 2)$
net1	0.68	0.24	0.04	0.04	0.12	0.68	0.16	0.04
net2	0.56	0.18	0.06	0.20	0.20	0.72	0.06	0.02
net3	0.78	0.08	0.04	0.10	0.22	0.64	0.06	0.08
net4	0.62	0.20	0.08	0.10	0.12	0.80	0.06	0.02
net5	0.66	0.22	0.02	0.10	0.24	0.60	0.04	0.12
net6	0.62	0.16	0.16	0.06	0.12	0.76	0.12	0.00
net7	0.68	0.08	0.00	0.24	0.24	0.68	0.00	0.08
net8	0.66	0.18	0.00	0.16	0.12	0.76	0.02	0.10
net9	0.42	0.16	0.06	0.36	0.20	0.70	0.04	0.06
net10	0.58	0.16	0.08	0.18	0.26	0.70	0.04	0.00
Mean	0.6260	0.1660	0.0540	0.1540	0.1840	0.7040	0.0600	0.0520
STD	0.0948	0.0525	0.0472	0.0962	0.0580	0.0595	0.0471	0.0424

Table A.22. Probabilities for Run 4 Cross Entropy

Net	$P(1 \mid 3)$	$P(2 \mid 3)$	$P(3 \mid 3)$	$P(4 \mid 3)$	$P(1 \mid 4)$	$P(2 \mid 4)$	$P(3 \mid 4)$	$P(4 \mid 4)$	$P(\text{good})$
net1	0.18	0.06	0.72	0.04	0.16	0.00	0.06	0.78	0.715
net2	0.08	0.16	0.74	0.02	0.12	0.08	0.08	0.72	0.685
net3	0.12	0.16	0.64	0.08	0.20	0.00	0.04	0.76	0.705
net4	0.14	0.10	0.70	0.06	0.14	0.00	0.06	0.80	0.730
net5	0.06	0.16	0.68	0.10	0.18	0.04	0.06	0.72	0.665
net6	0.02	0.16	0.76	0.06	0.08	0.06	0.04	0.82	0.740
net7	0.06	0.08	0.78	0.08	0.20	0.10	0.04	0.66	0.700
net8	0.10	0.12	0.72	0.06	0.18	0.02	0.06	0.74	0.720
net9	0.10	0.14	0.62	0.14	0.04	0.14	0.04	0.78	0.630
net10	0.20	0.12	0 60	0.08	0.16	0.10	0 02	0 72	0.650
Mean	0.1060	0.1260	0.6960	0.0720	0.1460	0.0540	0.0500	0.7500	0.6940
STD	0 0558	0.0366	0 0602	0 0329	0 0525	0 0499	0 0170	0 0474	0 0360

Appendix B. *Data File Samples and Processing Software*

B.1 Preprocessing of Correlation Product Data Files.

The preprocessing of the correlation product data was performed using a commercial digital signal processing package called DADiSP Worksheet^{tm}, by DSP Development Corporation, One Kendall Square, Cambridge, MA 02139. The software is a graphics-based worksheet with a multi-window environment. For this thesis, the 1,000 point data files were loaded into a DADiSP Labbook called THESIS which also contained the Worksheet called REDUCE2 shown below. The Command File shown on the next page was used to automate the reduction task. The Command File, loaded into WINDOW 1 of the Worksheet, controlled the input of the 1,000 point data files into the Worksheet and the output to drive A: of the reduced 50 point data files. A sample of a Direct Sequence correlation product before and after processing is shown in Figures B.1 and B.2.

```
DADiSP Worksheet algorithm implemented in a
worksheet called REDUCE2.

WINDOW 1 : <file read in here>
WINDOW 2 : Decimate(W1,2,1)
WINDOW 3 : Decimate(W1,2,2)
WINDOW 4 : Avg(W2,W3)
WINDOW 5 : Abs(W4) | fmax
WINDOW 6 : W4/getpt(W5,curpos(W5))  |  fmax  | nmove(-25)
WINDOW 7 : Extract(W6,curpos(W6),50)
```

Sample of DADiSP Worksheet Command File.

```
@cntl_home
@f8 CORR97.1 @cr @cr writea("A:\corrdat2\fcorr97.dat",w7) @cr
@f8 CORR98.1 @cr @cr writea("A:\corrdat2\fcorr98.dat",w7) @cr
@f8 CORR99.1 @cr @cr writea("A:\corrdat2\fcorr99.dat",w7) @cr
@f8 CORR100.1 @cr @cr writea("A:\corrdat2\fcorr100.dat",w7) @cr
@f8 CORR101.1 @cr @cr writea("A:\corrdat2\fcorr101.dat",w7) @cr
@f8 CORR102.1 @cr @cr writea("A:\corrdat2\fcorr102.dat",w7) @cr
@f8 CORR103.1 @cr @cr writea("A:\corrdat2\fcorr103.dat",w7) @cr
@f8 CORR104.1 @cr @cr writea("A:\corrdat2\fcorr104.dat",w7) @cr
@f8 CORR105.1 @cr @cr writea("A:\corrdat2\fcorr105.dat",w7) @cr
@f8 CORR106.1 @cr @cr writea("A:\corrdat2\fcorr106.dat",w7) @cr

     (Same pattern repeated for each CORRXX.1 file)

@f8 CORR264.1 @cr @cr writea("A:\corrdat2\fcorr264.dat",w7) @cr
@f8 CORR265.1 @cr @cr writea("A:\corrdat2\fcorr265.dat",w7) @cr
@f8 CORR266.1 @cr @cr writea("A:\corrdat2\fcorr266.dat",w7) @cr
@f8 CORR267.1 @cr @cr writea("A:\corrdat2\fcorr267.dat",w7) @cr
@f8 CORR268.1 @cr @cr writea("A:\corrdat2\fcorr268.dat",w7) @cr
@f8 CORR269.1 @cr @cr writea("A:\corrdat2\fcorr269.dat",w7) @cr
@f8 CORR270.1 @cr @cr writea("A:\corrdat2\fcorr270.dat",w7) @cr
@f8 CORR271.1 @cr @cr writea("A:\corrdat2\fcorr271.dat",w7) @cr
@f8 CORR272.1 @cr @cr writea("A:\corrdat2\fcorr272.dat",w7) @cr
@f8 CORR273.1 @cr @cr writea("A:\corrdat2\fcorr273.dat",w7) @cr
@esc @esc @esc y @esc @esc @esc
```

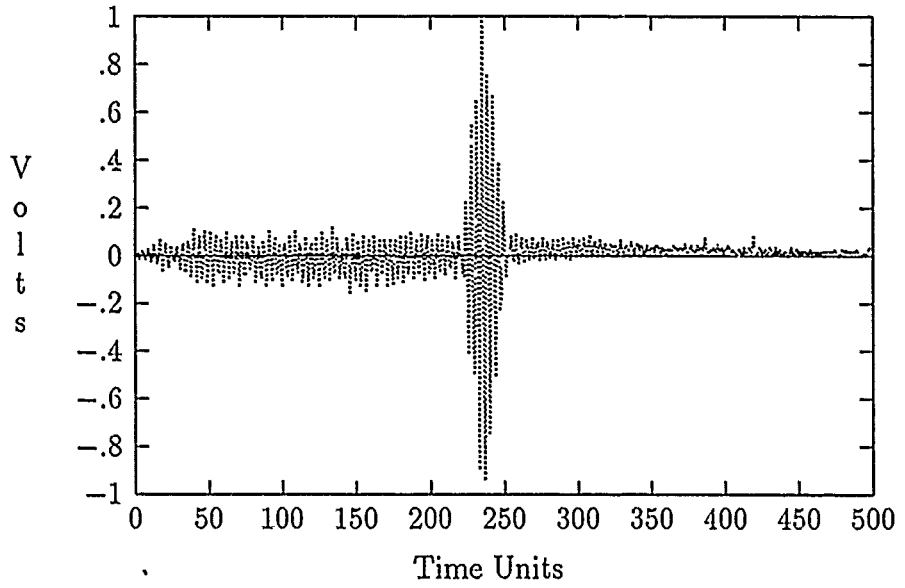

Figure B.1. Direct Sequence Correlation Product CORR18 Before Processing

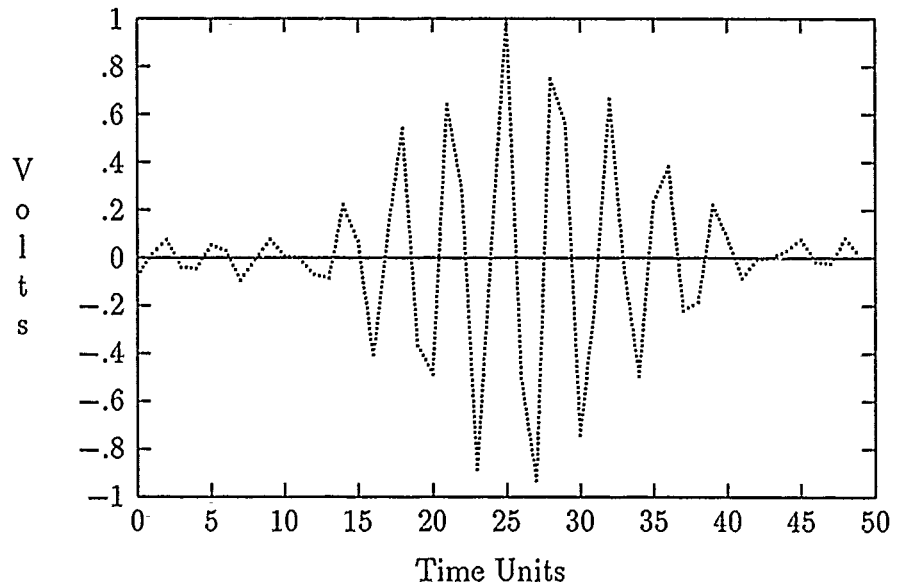

Figure B.2. Direct Sequence Correlation Product CORR18 After Processing

B.2 Construction of Datasets.

In this section, the details of the construction of the input data files required by the ANN simulator software will be presented. The data construction program is written in QuickBasic^{tm}. First, the program prompts the user for the name of an input file containing the sequence number, filename, and the class for each correlation signature to be used in the data set. Next, the user must provide a name for the file in which the output of this program will be written. This file will be used as the input to the ANN simulator. Finally, the user must specify the number of elements in each vector. This number must be 50 for all reduced correlation signatures used in this thesis. A sample of an input file and the source code for the program BUILDIN.BAS follows:

```
Sample input file for constucting a dataset.

1, "d:\data\corrdat1\fcorr6.dat", 1
2, "d:\data\corrdat4\fcorr61.dat", 2
3, "d:\data\corrdat1\fcorr9.dat", 1
4, "d:\data\corrdat4\fcorr63.dat", 2
5, "d:\data\corrdat1\fcorr12.dat", 1
6, "d:\data\corrdat4\fcorr65.dat", 2
7, "d:\data\corrdat1\fcorr14.dat", 1
8, "d:\data\corrdat4\fcorr67.dat", 2

(Same pattern repeated for each file included in dataset)

194, "d:\data\corrdat4\fcorr235.dat", 2
195, "d:\data\corrdat1\fcorr196.dat", 1
196, "d:\data\corrdat4\fcorr237.dat", 2
197, "d:\data\corrdat1\fcorr198.dat", 1
198, "d:\data\corrdat4\fcorr239.dat", 2
199, "d:\data\corrdat1\fcorr200.dat", 1
200, "d:\data\corrdat4\fcorr241.dat", 2
201, "d:\data\corrdat1\fcorr202.dat", 1
202, "d:\data\corrdat4\fcorr243.dat", 2
```

Source Code for BUILDIN

```
'Program: BUILDIN.BAS
'Description:  This routine reads a ASCII data file called
'names$ consisting of multiple lines of a sequence number,
'a filename, and a class number. The filenames contain 50
'element correlation product vectors in a column. Each vector
'file is read and then written to a data file named by the
'user. The format for the super file is :

'     element(0), element(1), ..., element(48), element(49)
'     Class #
'     element(0), element(1), ..., element(48), element(49)
'     Class #
'     element(0), element(1), ..., element(48), element(49)
'     Class #
'           .
'           .
'           .
'     EOF
'
'This file will be used as an input file for the ANN simulator
'software written by Dan Zahirniak.

INPUT "What is the file containing the name & class data"; names$
CLS
INPUT "What shall I name the ANN data file"; data$
CLS
INPUT "How many elements per vector"; inelements%
CLS
DIM vector!(inelements%)
PRINT "Name file - "; names$
PRINT "ANN data file - "; data$
PRINT
PRINT "File being processed - ";
OPEN super$ FOR OUTPUT AS #1
OPEN names$ FOR INPUT AS #2
DO UNTIL EOF(2)
   INPUT #2, number%, file$, class%
   LOCATE 4, 26
   PRINT file$
   OPEN file$ FOR INPUT AS #3
```

```
   FOR i = 0 TO (inelements% - 1)
      INPUT #3, vector!(i)
   NEXT i
   CLOSE #3
   FOR i = 0 TO (inelements% - 1)
      PRINT #1, vector!(i); " ";
   NEXT i
   PRINT #1,
   PRINT #1, class%
LOOP
CLOSE
END
```

B.3 ANN Simulator Menu and Output Files.

In this section, samples of the menu used to select the parameters for each network and samples of actual output files of the networks will be shown. The menu used to set-up the networks is contained in a larger file called NETMENU.C. The output files generated by the ANN simulator software will be used to construct the tables containing the results of the various runs.

The Menu for the ANN simulator is shown below. This is the menu used to run the CE back-propagation network with a randomization seed of 8.

```
/*****************TEST STUFF************************************/

     static char train_file[] = "class4.in";
     static char test_file[] = "class4.in";
     static char output_file[] · "Extra_info.out;
     static char selection_file[] = "CE4S8.SEL";
     static char MSE_file[] = "MSE_data.out";
     static char CFM_file[] = "CFM_data.out";
     static char CE_file[] = "CE4S8.OUT";

     normalize_the_data = 0;                    /* 1 = yes */
     find_the_distance = 0;                     /* 1 = yes */

     dimension = 50;
     train_set = 204;          /* Randomization Rule        */
     test_set = 200;           /* 1 - load separate files   */
     classes = 4;              /* 2 - load from single file */
     randomization_rule = 3;   /* 3 - load by class         */


     training_patterns_in_class[1] = 51;
     training_patterns_in_class[2] = 51;
     training_patterns_in_class[3] = 51;
     training_patterns_in_class[4] = 51;
     training_patterns_in_class[5] = 0;
     training_patterns_in_class[6] = 0;
     training_patterns_in_class[7] = 0;
     training_patterns_in_class[8] = 0;
```

```
  training_patterns_in_class[9] = 0;
  training_patterns_in_class[10] = 0;

  wght_seed= 0; sigma_seed= 0; data_seed= 8; record_seed= 1;

  network_type = 1; number_of_layers = 3;

  nodes_in_layer[0]=dimension;
  nodes_in_layer[1]=24;
  nodes_in_layer[2]=12;
  nodes_in_layer[3]=4;

  training_rule[0]=0;

  training_rule[1]=7;

/*1-nodes at data points 2-center class average 3-  K-means  */
/*sig-thres, out-thres   avg-thresh sigthresh   sig rule 3or4*/

/*4-kohonen  5-MSE backprop   6-CFM backprop  7-CE backprop  */
/*  nodes_x     MSE stuff        CFM stuff       CE stuff    */

  training_rule[2] = 0

/*1-matrix invert 2-MSE backprop 3-CFM backprop 4-CE backprop*/
/*5-Parzen window      MSE stuff      CFM stuff      CE stuff*/

  training_rule[3] = 0;

/*1-MSE backprop 2-CFM backprop 3-CE backprp 4-Parzen window */
/*  MSE stuff      CFM stuff       CE stuff                   */

  sigma_threshold = 4;            kohonen_iterations = 20000;
  output_threshold = 1;           nodes_x = 7;
  average_threshold = 1;

MSE_iterations =30000; CFM_alpha = 1.0;     CE_epsilon = .05;
MSE_error_delta = .1;  CFM_beta = 4.0;      CE_iterations=50000;
MSE_momentum = .1;     CFM_eta = .14;       CE_momentum = .05;
MSE_eta = .3;          CFM_zeta = 0;        CE_eta = 1.5;
MSE_successes = 100;   CFM_successes = 100; CE_successes =10000;
                       CFM_iterations = 50000;
```

```
                    CFM_momentum = .1;
                    CFM_delta = 1.0;


    transfer_function[0] = 0;      /* 1- sigmoidal */
    transfer_function[1] = 1;      /* 2 -rbf       */
    transfer_function[2] = 1;      /* 3- linear    */
    transfer_function[3] = 1;

   sigma_rule = 1;           /* Sigma rules 1- scale by constant */
   interference_threshold = .4;
   sigma_factor = .1;        /*     2 - half nearest neighbor    */
   sigma_constant = 1.0;     /*     3 - constant                 */
   p_neighbors = 7;          /*     4 - p neighbor average       */


/*****************************************************************/
```

The output files for the network selected by the menu shown on the previous pages will be shown. As specified in the first few lines of the menu, the output files shown will be called CE4S8.SEL and CE4S8.OUT. The .SEL file contains the information on the network set-up as well as output results such as classification accuracy and vectors misclassified. The .SEL files will be used to construct the probability matrix tables found in APPENDIX A. The classification accuracy percentages and the misclassified vectors are based on the "good" classification metric. The .OUT file, generated for back-propagation networks only, contain the training and test history of the networks based on the "right" classification metric. The .OUT files will be used to construct the back-propagation training performance plots found in Chapter 4.

```
    Sample .SEL file from ANN software simulator

    CE4S8.SEL

Training file = class4.in   Test file = class4.in
with 204  training vectors and 200  test vectors
dimension = 50 classes = 4
load by class
 training patterns in class 1 = 51
 training patterns in class 2 = 51
 training patterns in class 3 = 51
 training patterns in class 4 = 51

weight seed = 0  sigma seed = 0  data seed = 8  record seed = 1

starting network topology
 network type = feedforward with number of layers = 3
 nodes in layer 0 = 50
 nodes in layer 1 = 24
 nodes in layer 2 = 12
 nodes in layer 3 = 4

 layer 1 transfer function = sigmoid
```

```
 layer 2 transfer function = sigmoid
 layer 3 transfer function = sigmoid

CE layer 1 and all others
eplison = 0.050000 iterations = 50000 momentum = 0.050000
eta = 1.500000 errors = 15000

 Final topology
 network type = feedforward with number of layers = 3
 nodes in layer 0 = 50
 nodes in layer 1 = 24
 nodes in layer 2 = 12
 nodes in layer 3 = 4

training data
 total errors = 1
 per cent correct = 99.509804
 record 43 misclassified as 1
 records in class 1 = 51
 records in class 2 = 51
 records in class 3 = 51
 records in class 4 = 51
test data
 total errors = 56
 per cent correct = 72.000000
 record 232 misclassified as 2
 record 72 misclassified as 4
 record 209 misclassified as 1
 record 88 misclassified as 4
 record 47 misclassified as 1
 record 276 misclassified as 4
 record 4 misclassified as 4
 record 78 misclassified as 4
 record 395 misclassified as 1
 record 45 misclassified as 4
 record 55 misclassified as 1
 record 187 misclassified as 1
 record 137 misclassified as 4
 record 121 misclassified as 4
 record 0 misclassified as 2
 record 271 misclassified as 2
 record 212 misclassified as 4
```

```
record 255 misclassified as 1
record 53 misclassified as 1
record 161 misclassified as 1
record 264 misclassified as 2
record 171 misclassified as 3
record 70 misclassified as 4
record 92 misclassified as 2
record 285 misclassified as 4
record 259 misclassified as 3
record 146 misclassified as 2
record 322 misclassified as 2
record 64 misclassified as 2
record 125 misclassified as 1
record 208 misclassified as 2
record 48 misclassified as 4
record 197 misclassified as 1
record 103 misclassified as 1
record 370 misclassified as 1
record 369 misclassified as 4
record 119 misclassified as 3
record 275 misclassified as 1
record 38 misclassified as 2
record 313 misclassified as 3
record 374 misclassified as 2
record 378 misclassified as 1
record 318 misclassified as 2
record 14 misclassified as 4
record 101 misclassified as 1
record 80 misclassified as 2
record 247 misclassified as 1
record 332 misclassified as 2
record 143 misclassified as 1
record 142 misclassified as 2
record 82 misclassified as 1
record 280 misclassified as 4
record 118 misclassified as 1
record 66 misclassified as 1
record 36 misclassified as 2
record 296 misclassified as 4
records in class 1 = 50
records in class 2 = 50
records in class 3 = 50
```

```
records in class 4 = 50
total per cent correct = 85.891090
```

Sample .OUT history output file produced by ANN simulator

CE4S8.OUT

```
iteration = 1000 training correct = 0.000000  test correct = 0.00
iteration = 2000 training correct = 0.000000  test correct = 0.00
iteration = 3000 training correct = 2.450980  test correct = 2.50
iteration = 4000 training correct = 12.254902 test correct = 11.50
iteration = 5000 training correct = 15.196078 test correct = 12.50
iteration = 6000 training correct = 15.196078 test correct = 11.50
iteration = 7000 training correct = 17.647058 test correct = 12.50
iteration = 8000 training correct = 22.549019 test correct = 15.00
iteration = 9000 training correct = 29.411764 test correct = 18.50
iteration = 10000 training correct = 40.196079 test correct = 27.00
iteration = 11000 training correct = 37.745098 test correct = 24.50
iteration = 12000 training correct = 49.509804 test correct = 29.00
iteration = 13000 training correct = 53.921570 test correct = 36.50
iteration = 14000 training correct = 57.352940 test correct = 31.50
iteration = 15000 training correct = 53.431374 test correct = 36.50
iteration = 16000 training correct = 62.745098 test correct = 39.50
iteration = 17000 training correct = 38.725491 test correct = 29.50
iteration = 18000 training correct = 58.333332 test correct = 33.50
iteration = 19000 training correct = 72.058823 test correct = 42.50
iteration = 20000 training correct = 68.627449 test correct = 41.50
iteration = 21000 training correct = 61.764706 test correct = 35.00
iteration = 22000 training correct = 77.450981 test correct = 44.50
iteration = 23000 training correct = 75.490196 test correct = 45.00
iteration = 24000 training correct = 82.352943 test correct = 47.50
iteration = 25000 training correct = 80.392159 test correct = 46.00
iteration = 26000 training correct = 84.313728 test correct = 49.50
iteration = 27000 training correct = 82.843140 test correct = 44.50
iteration = 28000 training correct = 83.333336 test correct = 48.00
iteration = 29000 training correct = 88.235291 test correct = 49.00
iteration = 30000 training correct = 77.450981 test correct = 46.00
iteration = 31000 training correct = 83.823532 test correct = 52.00
iteration = 32000 training correct = 85.784317 test correct = 53.50
```

```
iteration = 33000 training correct = 81.372551 test correct = 47.00
iteration = 34000 training correct = 84.803925 test correct = 49.00
iteration = 35000 training correct = 86.274513 test correct = 52.00
iteration = 36000 training correct = 85.784317 test correct = 48.50
iteration = 37000 training correct = 75.980392 test correct = 53.00
iteration = 38000 training correct = 84.313728 test correct = 55.50
iteration = 39000 training correct = 86.764709 test correct = 54.50
iteration = 40000 training correct = 84.803925 test correct = 55.00
iteration = 41000 training correct = 87.254906 test correct = 54.50
iteration = 42000 training correct = 88.725487 test correct = 56.00
iteration = 43000 training correct = 89.705879 test correct = 57.00
iteration = 44000 training correct = 88.725487 test correct = 54.50
iteration = 45000 training correct = 89.215683 test correct = 55.00
iteration = 46000 training correct = 90.196075 test correct = 55.50
iteration = 47000 training correct = 86.274513 test correct = 56.50
iteration = 48000 training correct = 89.705879 test correct = 54.00
iteration = 49000 training correct = 88.725487 test correct = 57.50
iteration = 50000 training correct = 89.705879 test correct = 59.50
```

B.4 Processing of ANN Output.

In this section, the QuickBasic^{tm} programs used to process the information contained in the selection file generated by the ANN simulator will be presented. Both programs use the records misclassified to produce probability matrix tables located in Appendix A. PTAB2.BAS is used to produce the matrices for two class runs and PTAB4.BAS is used to produce the matrices for the four class run. The selection file as shown in Section B.3 of this appendix is edited to produce files to be used by the programs. The program requires the user to provide the name of a file containing a list of filenames. The files specified by the list contain records misclassified information taken from the selection file.

```
Sample input file for PTAB2.BAS containing the 8
records misclassified for a particular network

net1.dat

107
200
133
199
 23
 77
111
116
```

```
'Program: PTAB2.BAS
'This routine reads the data file containing test vectors
'misclassified yielded by each net for the two class runs. It
'then calculates the P matrix for that net. Multiplying the
'values of the P matrix by 100 yields the actual observed
'percent correct (or wrong) performance of the net.  The
'routine writes this info to a file in rows for each
'net data file in the name$ file. If the user specifies
```

```
'LOTUS format, the result is a table ready for import into
'LOTUS for computing the average of each column
'( P(1/1), P(2/1), P(1/2), P(2/2), and P(good) )



OPTION BASE 1
REM $DYNAMIC
LOCATE 23, 2
INPUT "What source file for the data file names"; name$
CLS
LOCATE 23, 2
INPUT "Filename for probability matrix table"; matrix$
CLS
LOCATE 23, 2
INPUT "Filename for out of class vector log"; verror$
CLS
DO
   LOCATE 23, 2
   INPUT "Do you want Lotus type file"; a$
LOOP UNTIL a$ = "y" OR a$ = "n"
CLS
LOCATE 12, 31
PRINT "WORKING ....."
DIM cerror%(2), vectornum%(50), classcount%(2)
DIM prob!(5)
OPEN matrix$ FOR OUTPUT AS #1
OPEN name$ FOR INPUT AS #2
OPEN verror$ FOR OUTPUT AS #3
IF a$ = "n" THEN
PRINT #1, "Net ID  P(1|1)   P(2|1)   P(1|2)   P(2|2)  P(good)"
PRINT #1, "-------------------------------------------------"
END IF
PRINT #3, "Net ID        Out - of - Class vectors"
PRINT #3, "-------------------------------------------------"
DO UNTIL EOF(2)
   FOR i = 1 TO 2
      cerror%(i) = 0
      classcount%(i) = 0
   NEXT i
   wrong% = 0
   INPUT #2, net$
```

```
OPEN net$ FOR INPUT AS #4
DO UNTIL EOF(4)
   INPUT #4, missed%
   missed% = missed% + 103
   IF missed% MOD 2 = 1 THEN
      classcount%(1) = classcount%(1) + 1
      cerror%(1) = cerror%(1) + 1
      wrong% = wrong% + 1
      vectornum%(wrong%) = missed%
   ELSE cerror%(2) = cerror%(2) + 1
      wrong% = wrong% + 1
      vectornum%(wrong%) = missed%
   END IF
LOOP
CLOSE #4
classcount%(1) = 50
classcount%(2) = 50
count% = classcount%(1) + classcount%(2)
prob!(2) = cerror%(1) / classcount%(1)
prob!(3) = cerror%(2) / classcount%(2)
prob!(1) = 1 - prob!(2)
prob!(4) = 1 - prob!(3)
prob!(5) = (classcount%(1) / count%) * prob!(1)
          + (classcount%(2) / count%) * prob!(4)
PRINT #3, LEFT$(net$, LEN(net$) - 4);
IF LEN(net$) - 4 = 4 THEN PRINT #3, SPC(4);
IF LEN(net$) - 4 = 5 THEN PRINT #3, SPC(3);
IF LEN(net$) - 4 = 6 THEN PRINT #3, SPC(2);
FOR i = 1 TO wrong%
   PRINT #3, vectornum%(i);
NEXT i
PRINT #3,
IF a$ = "n" THEN
   PRINT #1, LEFT$(net$, LEN(net$) - 4),
   FOR i = 1 TO 5
      PRINT #1, prob!(i); SPC(4);
   NEXT i
   PRINT #1,
ELSEIF a$ = "y" THEN
   PRINT #1, CHR$(34); LEFT$(net$, LEN(net$) - 4); CHR$(34);
   FOR i = 1 TO 5
      PRINT #1, ","; prob!(i);
```

```
      NEXT i
      PRINT #1,
   END IF
LOOP
CLS
CLOSE
END
```

Sample input file for PTAB4.BAS containing the 8
records misclassified and the incorrect classifications
for a particular network

net1.dat

```
107   1
200   2
133   1
199   3
 23   4
 77   3
111   1
116   4
```

```
'Program: PTAB4.BAS
'This routine reads the data file containing the records
'misclassified and the incorrect classifications yielded by a
'net on the test set vectors for the four class runs.
'It then calculates the P matrix
'for that net. Multiplying the values of the P matrix by 100
'yields the actual observed percent correct (or wrong)
'performance of the net.  The routine writes this
'info to a file in rows for each net data file in the name$
'file.  If the user specifies LOTUS format, the result is a
'table ready for import into LOTUS for computing the average
'of each column
```

```
OPTION BASE 1
REM $DYNAMIC
LOCATE 23, 2
INPUT "What source file for the data file names"; name$
CLS
LOCATE 23, 2
INPUT "Filename for correct probability matrix table"; matrix$
CLS
LOCATE 23, 2
INPUT "Filename for misclassified prob. matrix table"; matrixa$
CLS
LOCATE 23, 2
INPUT "Filename for out of class vector log"; verror$
CLS
DO
   LOCATE 23, 2
   INPUT "Do you want Lotus type file"; A$
LOOP UNTIL A$ = "y" OR A$ = "n"
CLS
LOCATE 12, 31
PRINT "WORKING ....."
DIM cerror%(17), vectornum%(400), classcount%(4)
DIM prob!(17)
OPEN matrix$ FOR OUTPUT AS #1
OPEN name$ FOR INPUT AS #2
OPEN verror$ FOR OUTPUT AS #3
OPEN matrixa$ FOR OUTPUT AS #5
PRINT #3, "Net ID        Out - of - Class vectors"
PRINT #3, "-----------------------------------------------"
DO UNTIL EOF(2)
   FOR i = 1 TO 17
      cerror%(i) = 0
   NEXT i
   wrong% = 0
   INPUT #2, net$
   OPEN net$ FOR INPUT AS #4
   DO UNTIL EOF(4)
      INPUT #4, missed%, called%
      missed% = missed% + 1
      IF missed% MOD 4 = 1 THEN
 IF called% = 2 THEN
```

```
   cerror%(6) = cerror%(6) + 1
ELSE
   IF called% = 3 THEN
      cerror%(7) = cerror%(7) + 1
   ELSE
      cerror%(8) = cerror%(8) + 1
   END IF
END IF
     ELSE
        IF missed% MOD 4 = 2 THEN
   IF called% = 1 THEN
      cerror%(9) = cerror%(9) + 1
   ELSE
      IF called% = 3 THEN
         cerror%(10) = cerror%(10) + 1
      ELSE
         cerror%(11) = cerror%(11) + 1
      END IF
   END IF
        ELSE
           IF missed% MOD 4 = 3 THEN
      IF called% = 1 THEN
         cerror%(12) = cerror%(12) + 1
      ELSE
         IF called% = 2 THEN
            cerror%(13) = cerror%(13) + 1
         ELSE
            cerror%(14) = cerror%(14) + 1
         END IF
      END IF
           ELSE
      IF called% = 1 THEN
         cerror%(15) = cerror%(15) + 1
      ELSE
         IF called% = 2 THEN
            cerror%(16) = cerror%(16) + 1
         ELSE
            cerror%(17) = cerror%(17) + 1
         END IF
      END IF
           END IF
        END IF
```

```
   END IF
wrong% = wrong% + 1
vectornum%(wrong%) = missed%
LOOP
CLOSE #4
classcount%(1) = 50
classcount%(2) = 50
classcount%(3) = 50
classcount%(4) = 50
count% = classcount%(1) + classcount%(2) + classcount%(3)
          + classcount%(4)
prob!(2) = cerror%(6) / classcount%(1)
prob!(3) = cerror%(7) / classcount%(1)
prob!(4) = cerror%(8) / classcount%(1)
prob!(1) = 1 - (prob!(6) + prob!(7) + prob!(8))
prob!(5) = cerror%(9) / classcount%(2)
prob!(7) = cerror%(10) / classcount%(2)
prob!(8) = cerror%(11) / classcount%(2)
prob!(6) = 1 - (prob!(9) + prob!(10) + prob!(11))
prob!(9) = cerror%(12) / classcount%(3)
prob!(10) = cerror%(13) / classcount%(3)
prob!(12) = cerror%(14) / classcount%(3)
prob!(11) = 1 - (prob!(12) + prob!(13) + prob!(14))
prob!(13) = cerror%(15) / classcount%(4)
prob!(14) = cerror%(16) / classcount%(4)
prob!(15) = cerror%(17) / classcount%(4)
prob!(16) = 1 - (prob!(15) + prob!(16) + prob!(17))
A = (classcount%(1) / count%) * prob!(1)
B = (classcount%(2) / count%) * prob!(2)
C = (classcount%(3) / count%) * prob!(3)
D = (classcount%(4) / count%) * prob!(4)
prob!(17) = A + B + C + D
PRINT #3, LEFT$(net$, LEN(net$) - 4);
IF LEN(net$) - 4 = 4 THEN PRINT #3, SPC(4);
IF LEN(net$) - 4 = 5 THEN PRINT #3, SPC(3);
IF LEN(net$) - 4 = 6 THEN PRINT #3, SPC(2);
FOR i = 1 TO wrong%
   PRINT #3, vectornum%(i);
NEXT i
PRINT #3,
IF A$ = "n" THEN
   PRINT #1, LEFT$(net$, LEN(net$) - 4),
```

```
      FOR i = 1 TO 8
 PRINT #1, prob!(i); SPC(4);
      NEXT i
      PRINT #1,
   ELSEIF A$ = "y" THEN
      PRINT #1, CHR$(34); LEFT$(net$, LEN(net$) - 4); CHR$(34);
      FOR i = 1 TO 8
 PRINT #1, ","; prob!(i);
      NEXT i
      PRINT #1,
   END IF
   IF A$ = "n" THEN
      PRINT #5, LEFT$(net$, LEN(net$) - 4),
      FOR i = 9 TO 17
 PRINT #5, prob!(i); SPC(1);
      NEXT i
      PRINT #5,
   ELSEIF A$ = "y" THEN
      PRINT #5, CHR$(34); LEFT$(net$, LEN(net$) - 4); CHR$(34);
      FOR i = 9 TO 17
 PRINT #5, ","; prob!(i);
      NEXT i
      PRINT #5,
   END IF
LOOP
CLS
CLOSE
END
```

Bibliography

1. DeBerry, John W. "Classification of Acousto-Optic Correlation Signatures of Spread Spectrum Signals Using Artificial Neural Networks," MS thesis, AFIT/GE/ENG/89D-10. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1989 (AD-A215045).
2. Gorman, R. P. and T. J. Sejnowski. "Learned Classification of Sonar Targets Using a Massively Parallel Network," *IEEE Tranactions on Acoustics, Speech, and Signal Processing, vol. 36*: 1135–1140 (July 1988).
3. Lippman, R. P. "An Introduction to Computing with Neural Nets," IEEE ASSP Magazine: 4–22 (April 1987).
4. Lippman, R. P. "Pattern Classification Using Neural Networks," *IEEE Communications Magazine*: 47–63 (November 1989).
5. Moody, John and Christian Darken. "Learning with Localized Receptive Fields," *Proceedings of the 1988 Connectionist Models Summer School*: 133–143, New Haven CT: Yale Computer Science (1988).
6. Norman, D. M. Spread Spectrum Communications handout distributed in EENG 667, Coding and Information Theory. School of Engineering, Air Force Institute of Technology (AU), Wright Patterson AFB OH, March 1990.
7. Nowlan, Steven J. *Max Likelihood Competition in RBF Networks.* Technical Report CRG-TR-90-2, Toronto Canada: Department of Computer Science, University of Toronto (February 1990).
8. Pickholtz, Raymond L. *et al.* "Theory of Spread-Spectrum Communication – A Tutorial," *IEEE Transactions on Communications, vol. 30*: 855–884 (May 1982).
9. Renals, S. and Richard Rohwer. "Phoneme Classification Experiments Using Radial Basis Functions," *Proceedings of the International Joint Conference on Neural Networks*: 461–467 (1989).
10. Rogers,Steven K. *et al.* "An Introduction to Biological and Artificial Neural Networks." wpafb: afit, 1990.
11. Rummelhart, David E. *et al.* "Parallel Distributed Processing." Cambridge MA: The MIT Press, 1986.
12. Tamura, S. and A. Waibel. "Noise Reduction Using Connectionist Models," *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, paper 512.7*: 553-556 (1988).

13. Tarr, G. L. "Dynamic Analysis of Feedforward Neural Networks Using Simulated and Measured Data," MS thesis, AFIT/GE/ENG/88D-54. School of Engineering, Air Force Institute of Technology (AU), Wright Patterson AFB OH, December 1988 (AD-A202573).

14. Troxel, S. E. *et al.* "The Use of Neural Networks in PSRI Target Recognition," *IEEE International Conference on Neural Networks*: 593–600 (1988).

15. Waibel, Alexander H. and John B. Hampshire. "A Novel Objective Function for Improved Phoneme Recognition," *IEEE Transactions on Neural Networks, vol. 1*: 216–227 (June 1990).

16. Waibel, Alexander H. *et al.* "Phoneme Recognition Using Time-Delay Neural Networks," *IEEE Transactions on Acoustics, Speech, Signal Processing, vol. ASSP-37*: 328–339 (March 1989).

17. Zahirniak, Dan. "Characterization of Radar Signals Using Neural Networks," MS thesis, AFIT/GE/ENG/90D-69. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1990.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1990	Master's Thesis

4. TITLE AND SUBTITLE

CLASSIFICATION OF CORRELATION SIGNATURES OF SPREAD SPECTRUM SIGNALS USING NEURAL NETWORKS

5. FUNDING NUMBERS

6. AUTHOR(S)

Richard A. Chapman, GS-12

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology, WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GE/ENG/90D-11

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

US Army Harry Diamond Labs
SLC-HD-ST-OP
2800 Powder Mill Road
Adelphi MD 20783

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The major goals of this research were to determine if Artificial Neural Networks (ANNs) could be trained to classify the correlation signatures of two classes and four classes of spread spectrum signals. Also, the possibility of training an ANN to classify features of the signatures other than signal class was investigated. Radial Basis Function (RBF) and Back-Propagation Networks were used for the classification problems. Correlation signatures of four types or classes were obtained from United States Army Harry Diamond Laboratories. The four types are as follows: direct sequence (DS), linearly-stepped frequency hopped (LSFH), randomly-driven frequency hopped (RDFH), and a hybrid of direct sequence and randomly-driven frequency hopped (HYB). These signatures were preprocessed and separated into various data sets for presentation to the ANNs. RBF and Back-Propagation Networks trained directly on two classes (DS and LSFH) and four classes (DS, LSFH, RDFH, and HYB) of correlation signatures. Classification accuracies ranged from 79% to 92% for the two class problem and from 70% to 76% for the four class problem. The RBF Networks consistently produced accuracies from 5% to 10% higher than accuracies produced by Back-Propagation Networks. Also, the RBF Networks required significantly less training time for all cases. In attempts to classify the signatures by parameters other than signal type, the results were inconclusive regarding the usefulness of ANNs.

14. SUBJECT TERMS

Neural Networks, Spread-Spectrum, Signal Classification, Pattern Recognition, Backward Error Propagation, Radial Basis Function

15. NUMBER OF PAGES

133

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

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to ***stay within the lines*** to meet ***optical scanning requirements.***

Block 1. Agency Use Only *(Leave blank)*

Block 2. Report Date. Full publication date including day, month, and year, if available (e g 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	- Contract	PR	- Project
G	- Grant	TA	- Task
PE	- Program Element	WU	- Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. *(If known)*

Block 11. Supplementary Notes. Enter information not included elsewhere such as. Prepared in cooperation with..., Trans. of.. , To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

- DOD - See DoDD 5230.24, "Distribution Statements on Technical Documents."
- DOE - See authorities.
- NASA - See Handbook NHB 2200.2.
- NTIS - Leave blank.

Block 12b. Distribution Code.

- DOD - Leave blank.
- DOE - Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
- NASA - Leave blank.
- NTIS - Leave blank.

Block 13. Abstract. Include a brief *(Maximum 200 words)* factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code *(NTIS only)*

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.