AN ANALYSIS OF STOPPING CRITERIA IN ARTIFICIAL NEURAL NETWORKS

THESIS

Bruce Kostal
Captain, USAF

AFIT/GST/ENS/94M-07

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

Wright-Patterson Air Force Base, Ohio

1

AFIT/GST/ENS/94M-07

AN ANALYSIS OF STOPPING CRITERIA IN ARTIFICIAL NEURAL NETWORKS

THESIS
Bruce Kostal
Captain, USAF

AFIT/GST/ENS/94M-07

Approved for public release; distributi unlimited

AFIT/GST/ENS/94M-07

AN ANALYSIS OF STOPPING CRITERIA IN ARTIFICIAL NEURAL NETWORKS

THESIS

Presented to the Faculty of the Graduate 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 Operations Research

Bruce Kostal, B.S., M.B.A.
Captain, USAF

March, 1994

THESIS APPROVAL

STUDENT: Bruce E. Kostal, USAF CLASS: GST-94M

THESIS TITLE: AN ANALYSIS OF STOPPING CRITERIA IN ARTIFICIAL NEURAL NETWORKS

DEFENSE DATE: 2 March 94

COMMITTEE:

Name/Title/Department	Signature
Advisor: Kenneth W. Bauer, Lt Col, USAF Associate Professor of Operations Research Department of Operational Sciences	Kenneth W Bauer
Reader: James S. Shedden, Lt Col, USAF Assistant Professor of Operations Research Department of Operational Sciences	James S Shedden

Preface

This study examines artificial neural networks. In particular, I attempted to find a better artificial neural network (ANN) by determining the optimal point at which to terminate training. The idea for the study was recommended to me by my advisor, Lt Col Ken Bauer, and two of his doctoral students, Capt Jean Steppe and Capt Lisa Belue. The results of this study provide useful information regarding determining when to stop ANN training and deciding upon a final set of ANN weights. I would like to thank several people for their contributions to this thesis effort. Lt Col Bauer has spent many hours guiding and inspiring me after my original idea for a topic fell through. Lt Col Jim Shedden served as a reader for my thesis and provided an independent viewpoint and insight into my written effort. Capt Steppe, Capt Belue, and Capt Greg Reinhart have provided valuable knowledge on artificial neural networks, going beyond the information found in the literature.

Perhaps the greatest thanks goes to my family. First of all, the greatest thanks goes to my wife Donna who managed my life for the past eighteen months. Next, to my daughter Kara who tolerated my absences and dutifully told me to go study when she did not want me around. And finally to Whitney who, by coming into this world on 20 February 1994, provided me the incentive to work hard to finish this thesis so I would be in a position to witness her arrival.

Bruce Kostal

Table of Contents

Page

Page

List of Figures

Figure Page

List of Tables

AFIT/GST/ENS/94M-07

Abstract

The goal of this study was to decide when to terminate training of an artificial neural network (ANN). In pursuit of this goal, several characteristics of the ANN were monitored throughout ANN training: classification error rate (of the training set, testing set, or a weighted average of the two); moving average classification error rate; measurements of the difference between ANN output and desired output (error sum of squares, total absolute error, or largest absolute error); or ANN weight changes (absolute weight change, squared weight change, or relative weight cnange). Throughout this research, the learning rate was held constant at 0.35. The momentum was not used because the primary interest of this study was evaluating when to terminate training as opposed to the speed of reaching a decision. The ANN structure was held constant with two input features, two output classes, and one hidden layer. Finally, the practice of training after processing each exemplar was followed. Results indicated three conclusions. First, multiple runs are required for ANN analysis because ANNs are not guaranteed to converge to the same solution. Second, the classification error rate, a moving average of the classification error rate, and the total absolute error (all computed on the testing set) provided a similar final classification. Third, once the stopping criteria functions cease to decrease, select a set of ANN final weights at random. On average, they yielded as low as or lower classification error rate and variance of the classification error rate than other possible choices.

AN ANALYSIS OF STOPPING CRITERIA IN ARTIFICIAL NEURAL NETWORKS

I. Introduction

1.1 Background

The idea of a computer or machine controlling our lives has been around for a long time. Until recently, machines could only do what they were told (which, due to some programming errors, may not have been what we wanted them to do). Now, however, through artificial intelligence and artificial neural networks, we are attempting to perform much more complicated tasks with computers. This paper will explore a method to improve an artificial neural network, or ANN. In particular, the study attempts to find a better way to determine the appropriate time to terminate ANN training.

Artificial neural networks are a relatively new area of research. In general terms, an ANN is a mathematical function which 'learns', thereby mimicking the work of the human brain. By analyzing training data, where the output is known, the ANN assigns weights to functions to 'learn' from its past experience. Following this iterative scheme, the network learns from training data, continually improving in its ability to predict the answer to a 'yes-no' question. Next, the ANN moves to a separate set of testing data to determine if the weights are correct. The goal of this continued improvement is the ability to predict a result based upon independent input factors.

Artificial intelligence and artificial neural networks are different. Guyon describes this difference as follows: "The main differences between the classical approach of Artificial Intelligence (AI) and Neuron Networks (NN) is that AI requires considerable detailed programming whereas NN rely heavily on learning" (Guyon, 1991: 243).

1.2 Problem Statement

An ANN currently has the capability to improve in its ability to predict an output based upon inputs. However, it is extremely difficult to determine when this improvement ceases to occur and an overlearning of the training data begins. This study measured the classification error rate of a separate validation set to search for a criterion which best predicted the point at which to terminate ANN training.

1.3 Scope

The goal of this study was to perform artificial neural network (ANN) analysis. The ANN algorithm used was the back propagation algorithm. The study used classification error rate of the validation set as a measurement standard to evaluate several proposed stopping criteria. Each proposed stopping criterion had a resulting output function which changed during ANN training. The study compared these stopping criteria functions with the classification error rate of an independent validation set which was assumed to be the best representation of the true population. The stopping criterion which had the lowest error classification rate was judged to be best. Then, this best stopping criterion was compared to more commonly used stopping criteria measures involving the classification error rate. The goal was to find a more automated and less arbitrary method of selecting the optimum or near optimum point at which to terminate ANN training.

As part of the research, decisions were made regarding methods and variables within the ANN. As Nelson notes, the tasking of an artificial neural net programmer is different from that of a normal computer programmer. The ANN programmer's task is to "specify transfer functions (equations to determine thresholds), training laws (rules that set initial weights and equations that modify weights), and the structure of the network (number of nodes, layers, and interconnections)". Additionally, the programmer must decide "if and how often the processing elements would update (continuously or periodically)" (Nelson, 1991: 55). Within this context, the study involved programming to search for an optimal stopping criterion or criteria.

1.4 Overview

This paper begins with a review of the literature found in Chapter 2, in which ANNs and eleven potential stopping criteria are discussed. The study continues with an explanation of the methodology which was used (Chapter 3). Additionally, Chapter 3 details the Exclusive OR (XOR) problem and the meshed data set problem which were used for this study. Chapter 4 gives the results of ANN experiments on the XOR and the meshed data set problems. Finally, Chapter 5 outlines the general procedure developed and refined in this study and lists areas of future research.

II. Literature Review

This chapter provides a literature review of artificial neural networks (ANN), with particular emphasis on the area of this study. Specifically, it will cover general information, building blocks of the back propagation algorithm, the back propagation algorithm itself, and close by covering the area of this study, criteria for determining when to stop training the artificial neural network.

2.1 General Information

To begin, the reader needs a general picture of the artificial neural network. An ANN is a family of algorithms used in learning which are based upon concepts developed from observing how the human brain works (Müller, 1990: 12). They use parallel processing to look at the entire picture instead of the processing each input individually. Nelson provides an example of this parallel processing which is reproduced in Figure 1. If each input is individually processed as in the traditional computer, that is, one letter or information bit at a time, the saying would be extremely difficult to read. For example, it would be virtually impossible to even know that the figure contained three separate sayings if the information were processed in this piecemeal manner. By stepping back and processing more than one input at a

_ ST_TCH _N
T_M_ S_V_S
N_N_

_ P_NNY S_V_D
_S _ P_NNY
__RN_D

D_N'T THR_W _W_Y
TH_ _LD B_CK_T
_NT_L Y__ KN_W
WH_TH_R TH_ N_W
_N_ H_LDS W_T_R.

Figure 1. Missing Letter Example

time, however, the true picture becomes clear. The reader can see that the vowels are missing and quickly fill in the missing letters to come up with the three sayings (Nelson, 1991: 15 - 16). The ANN tries to do the same thing; it takes more than one input and puts those inputs together to learn and provide correct classifications.

Artificial neural networks go by many names, such as the following: neural networks, parallel distributed processing models, connectivist or connectionism models, adaptive systems, self-organizing systems, neurocomputing, or neuromorphic systems (Nelson, 1991: 19). Each name may mean something slightly different. This paper will use the term artificial neural network or ANN.

In his overview, Lippmann presents a taxonomy of six types of ANNs. Lippmann's diagram is reproduced in Figure 2. This taxonomy is divided based upon

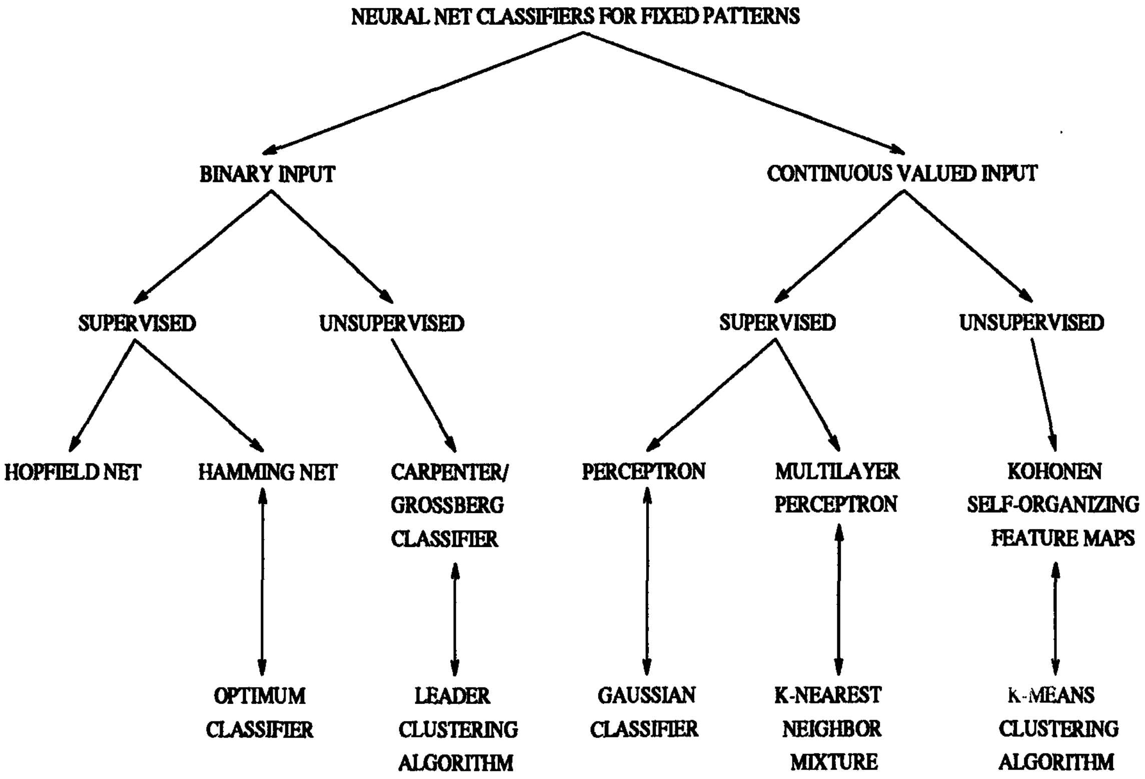

NEURAL NET CLASSIFIERS FOR FIXED PATTERNS
BINARY INPUT
CONTINUOUS VALUED INPUT
SUPERVISED
UNSUPERVISED
SUPERVISED
UNSUPERVISED
HOPFIELD NET
HAMMING NET
CARPENTER/ GROSSBERG CLASSIFIER
PERCEPTRON
MULTILAYER PERCEPTRON
KOHONEN SELF-ORGANIZING FEATURE MAPS
OPTIMUM CLASSIFIER
LEADER CLUSTERING ALGORITHM
GAUSSIAN CLASSIFIER
K-NEAREST NEIGHBOR MIXTURE
K-MEANS CLUSTERING ALGORITHM

Figure 2. Types of Artificial Neural Networks

the method used by the neural network. The first division is based upon whether the input data is binary or continuous. The continuous algorithms can also be used with binary input data. The second dividing test is based upon whether or not the training is supervised. Finally, the supervised training is again split (Lippmann, 1987: 6-7). This research will use continuous input, supervised training, and multiple layers.

Before going further into neural networks, it is best to define the terms which follow:

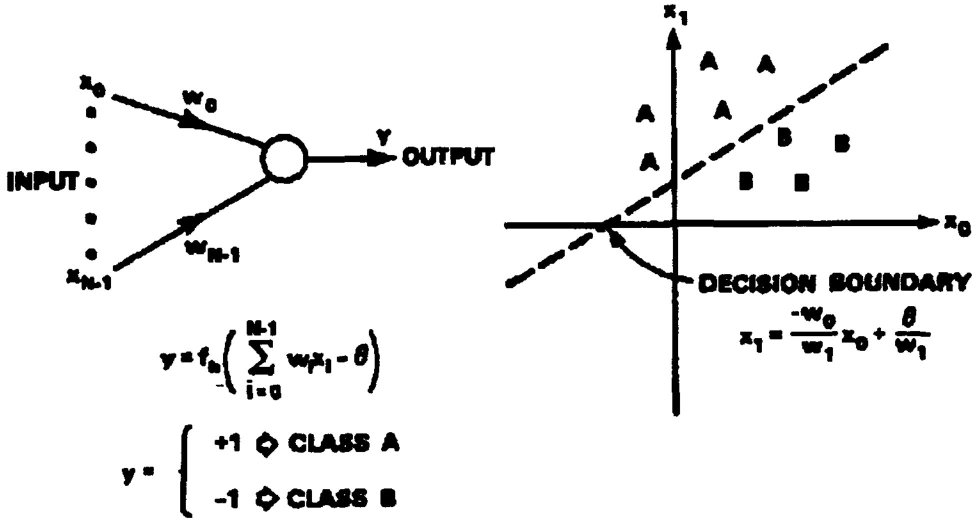
- Epoch: A cycle of presenting all training cases one time (Weiss, 1991: 85).
- Exemplars: Example data points for which the output class is known. The network is trained using these data points (Tou, 1974).
- Multilayer perceptrons: "[Feedforward] nets with one or more layers of nodes between the input and output nodes." The additional layers are where the hidden layers reside (Lippmann, 1991: 15).
- Feature: An input attribute of the output to be measured which contains useful information for distinguishing an item between the various classes. A feature is similar to an independent variable (Tou, 1974).
- Feedforward network: This type of network exists when no "processing element output can be an input for a node on the same layer or a preceding layer" (Nelson, 1991: 50-51). A feedforward network is the usual type of artificial neural network found in the literature.
- Hidden Units: Neurons which are neither input or output units. They are located between the input and output units in a multilayer neural network (Guyon, 1991: 224).
- Training Procedure: The way a neural network 'learns'. This will be covered in greater detail later.

2.2 *Artificial Neural Network Building Blocks*

Before covering the specific steps to be taken in the back propagation algorithm, it is first necessary to understand some of the building blocks that the algorithm will use. These building blocks are covered in this section.

2.2.1 Single Layer Perceptron. The simplest form of artificial neural network is the single layer perceptron. It is characterized by weights and features (independent input variables) (Weiss, 1991: 84).

A single layer perceptron can be represented as the weighting scheme or evaluation criterion which allows one to draw a dividing line in a two-dimensional space which segregates two types of outputs into separate, distinct regions. For an example, see Figure 3 which is reproduced from Lippmann. In Figure 3, there are two classes of outputs (A and B). These outputs can be separated by the line in the figure. After this line is drawn, one need only know on which side of the line a point is to determine whether it is type A or type B. Using a learning scheme, a line can be drawn to separate A and B based upon the input features, which would be the



x_0
w_0
INPUT
x_{N-1}
w_{N-1}
y
OUTPUT
$y = f_h\left(\sum_{i=0}^{N-1} w_i x_i - \theta\right)$
$y = \begin{cases} +1 \Rightarrow \text{CLASS A} \\ -1 \Rightarrow \text{CLASS B} \end{cases}$
x_1
A
A
A
A
A
B
B
B
B
x_0
DECISION BOUNDARY
$x_1 = \frac{-w_0}{w_1} x_0 + \frac{\theta}{w_1}$

Figure 3. Single Layer Perceptron

values of x_0 and x_1 here. In multidimensional space, a hyperplane divides the region (Lippmann, 1987: 13).

The previous example (Figure 3) is linearly separable, meaning a line can be drawn between the two classes. "When the two classes are linearly separable, the perceptron training procedure will be guaranteed to converge to an answer" (Weiss, 1991: 85). (The perceptron training procedure uses the back propagation algorithm which will be explained in Section 2-3.) However, if the classes are not linearly separable, "the procedure will not converge and will keep on cycling through the data forever" (Weiss, 1991: 87). Linear separability is the exception rather than the rule. Minsky and Papert illustrated this weakness of the perceptron. They pointed out that if the individual classes investigated under the single perceptron algorithm method can not be separated by a hyperplane, then the single perceptron algorithm is not appropriate (Lippmann, 1987: 15).

Training or adjustment of the weights occurs in the single layer perceptron only when an error (misclassification) occurs. "When a false positive error occurs, i.e., the true answer is 0 and the perceptron says 1, each weight is adjusted by subtracting the corresponding value in the input pattern, and 1 is subtracted from the threshold. For false negatives, each weight is adjusted by adding the corresponding value in the input pattern, and 1 is added to the threshold. The hope is that each adjustment will move the weights closer to the true weights" (Weiss, 1991: 85).

2.2.2 Multilayer Perceptron. Multilayer perceptrons are more flexible than the single layer perceptrons. Single layer perceptrons are only capable of dividing a space with one hyperplane. Multilayer perceptrons, on the other hand, can "form any, possibly unbounded, convex region in the space spanned by the inputs." The convex regions formed in the multilayer perceptron can have at most the number of sides as there are nodes in the first layer of the network (Lippmann, 1987: 16).

Weiss deals with either one or two hidden layers in the multilayer network. With one hidden layer, he notes that the multilayer network can "implement most decision surfaces, and can closely approximate any decision surface (Weiss, 1991: 94). Regarding the three layer network which includes two hidden layers, Weiss lists its attributes as being able to "implement any separating decision surface when sufficient hidden units are represented in the two layers" (Weiss, 1991: 94). While Weiss compliments the more capable network with more than one hidden layer, he concludes that one hidden layer is sufficient and that "additional layers do not add any representational power to the discrimination" (Weiss, 1991: 94). This paper concentrates on the case where one hidden layer is used.

2.2.3 The Training Process. Applications of neural networks involve four levels of learning: No learning (fixed weights), supervised learning, semi-supervised learning, and unsupervised learning (Guyon, 1991: 233). Supervised learning involves minimizing some cost function E which accumulates the errors as measured by the difference between the actual outputs and the desired outputs (Guyon, 224). (The function E is an error function tracking the difference between the ANN output and the desired output. The actual form of E varies. It may be represented by the absolute value of the error, the squared error, or some other form. Forms of the function E to be investigated will be explained later. For the mathematical formulation of the various functions E, please see Appendix B.) The goal is to minimize this function E. Guyon lists two techniques for this minimization: Monte Carlo techniques and gradient descent. Guyon prefers gradient descent because, although it usually leads to a suboptimal solution (or a local maximum or minimum), it is computationally less intensive than the Monte Carlo techniques (such as simulated annealing) (Guyon, 1991: 224). Supervised learning and gradient descent techniques were used in this study. Additionally, procedures exist to compensate for the gradient descent's propensity to find suboptimal solutions. These procedures will be covered later.

Generally, there are three versions of algorithms used to train a neural network: Hebb's rule, the perceptron algorithm, and the Widrow-Hoff algorithm. These latter two algorithms are basically the same; the difference being that, under the perceptron, training is only done if the network gives the wrong answer. Under the Widrow-Hoff algorithm, training is done as long as the answer is not perfect (Guyon, 1991: 220 - 221). This study exclusively used the Widrow-Hoff algorithm. The mathematical rules for this algorithm will be covered later in the back propagation section.

To conduct the training process, the data is split into two groups. The first group is called the training set. The training set is used "to design the classifier" (Weiss, 1991: 26). This means that the training data is given to the artificial neural network and weights are adjusted, rewarding correct responses and penalizing incorrect responses. The second group of data is called the testing set. The testing set is not given to the ANN until the training set has reached a classifying decision. After this decision is reached, the testing set is given to the ANN and an error rate (or some other measure such as error sum of squares or total absolute error) is determined. This error rate of the artificial neural network on the testing set is known as the test sample error rate (Weiss, 1991: 28). Additionally, data can be divided a third time into a set called the validation set which can be used after training and testing are complete (Wiggins, 1991: 28).

Essentially, the learning or training rules specify an "initial set of weights and indicate how the weights should be adapted during use to improve performance" (Lippmann, 1987: 4). The key to this learning process is the adjustment of the weights. To begin with, the artificial neural network is given the training data. Weights which lead to the ANN correctly identifying the response are strengthened and weights which lead to incorrect responses are weakened. The key here is getting the response trained to fall on the correct side of the threshold level (Nelson, 1991: 48).

Weiss addresses the concept of sufficiency of data. He states a "widely used heuristic" that "a network should average at least ten samples per weight in the network" (Weiss, 1991: 104). Additionally, Weiss concludes that larger sample sizes (of 1000, 5000, or higher) yield very accurate results. On the other hand, he notes that the lack of data is more often the limiting factor. Given this potential dearth of data, Weiss recommends splitting the data in the following proportions: $\frac{2}{3}$ for the training set and $\frac{1}{3}$ for the testing set (Weiss, 1991: 28 - 30).

2.2.4 Error Rate. In the absence of linear separability, errors will occur in the artificial neural network learning process. Throughout the process, then, Weiss notes that the goal could also be stated as "finding the best fit to the sample data without overspecializing the learning system" (Weiss, 1991: 37).

When developing a learning system, the apparent error rate of the system developed from a sample (the classification error rate of the training set) is the 'obvious' place to begin. However, Weiss notes that the apparent error rate is "a poor estimator of futu.e performance." Weiss continues "In general, apparent error rates tend to be biased optimistically. The true error rate is almost invariably higher than the apparent error rate. This happens when the classifier has been overfitted (or overspecialized) to the particular characteristics of the sample data" (Weiss, 1991: 24). The best way to correct for this is to ensure a random sample of data has been obtained. This maximizes the opportunity for finding the true error rate. Regarding this quest, Weiss noted that "humans have difficulty doing things randomly" (Weiss, 1991: 26). Therefore, using a machine or computer to decide upon the random sample is recommended (Weiss, 1991: 26).

Weiss notes that resampling methods can provide an even better estimate of the true error rate, especially when the data set is small. A simple form of resampling technique is called the leaving-one-out technique. In it, the entire sample is treated as part of the training set except for one data point which is treated as the testing set. Each data point is iteratively rotated into the status of the testing set. Because of the extensive number of iterations required under this technique (one iteration for each exemplar), Weiss recommends using it only on smaller sample sizes. With the continuing evolution and improvement of computer design and speed, however, the extensive number of iterations required is less of a problem than it once was. The leaving-one-out technique is also noteworthy because it is unbiased and accurate. However, it has a weakness of a relatively high variance, especially in smaller samples (Weiss, 1991: 30 - 33). As an alternative, Weiss mentions the bootstrapping technique (Weiss, 1991: 33 - 36).

Resampling methods such as the leaving-one-out technique or bootstrapping can be used for other statistical parameters besides estimating error rate. Weiss mentions that they can be used for evaluating how many variables to put in the model and evaluating how many nodes to put in the decision trees (Weiss, 1991: 37).

Since the particular resampling technique depends upon the sample size, Weiss offers guidance. Use cross-validation if the sample size is greater than 100. Use leaving-one-out if the sample size is less than 100. Weiss further states that, in the situation of 'very small samples' (which he defines as fewer than 50 cases), bootstrapping techniques "may be computed" (Weiss, 1991: 38).

Weiss points out that adding additional features, unlike in regression, may actually yield worse results. He lists two possible reasons. First, the predictive ability of some classifiers may decrease with poor or noisy data. Second, some methods may overweigh redundant features which essentially measure the same thing. The effect here is that of counting something twice. Since actual data is often noisy or redundant, the impact of adding weak features to the data set may be to degrade performance. To limit this impact, feature selection is used to minimize the effect of noise and redundancy. The effect of feature selection is to throw out or eliminate features which do not have positive impact upon the predictive capability of the classification scheme (Weiss, 1991: 40).

The true error curve is a function of the complexity of the model. As complexity increases, the true error rate initially decreases before leveling off. After this flattening, it wanders around for a period before increasing. Using this fact, Weiss concludes that, if the error rates are close, one should select the simpler model. He expands this to explain the one-standard error heuristic, which "selects the simplest solution that falls within one standard error of the minimum error rate solution" (Weiss, 1991: 45 - 46).

2.2.5 The Sigmoid Function.

The sigmoid function is another tool used in neural networks which must be understood before discussing the actual processing algorithm. It is used as a 'limiter,' or something that distinguishes between the images or desired responses. There are three types of limiters: hard limiters, threshold logic elements, and sigmoidal nonlinearities (Lippmann, 1987: 4). See Figure 4, reproduced from Lippmann.

Recently, the sigmoid unit, a modified version of the McCulloch-Pitts neuron, has increased in popularity. The sigmoid unit uses the smoother hyperbolic tangent function which has the advantage of being a differentiable function, making it possible to use gradient search techniques for training the multilayer neural network (Guyon, 1991: 223). Nelson stresses the advantage of the sigmoid function by saying that "both the function and its derivatives are continuous" (Nelson, 1991: 48).

Nelson lists two sigmoid transfer functions. In both cases below, y is the output of the sigmoid function and x, the input, is any real number. First, if the transfer function uses outputs of 0 and 1, the transfer sigmoid is defined as follows:

$$y = \frac{1}{1 + e^{-x}}$$

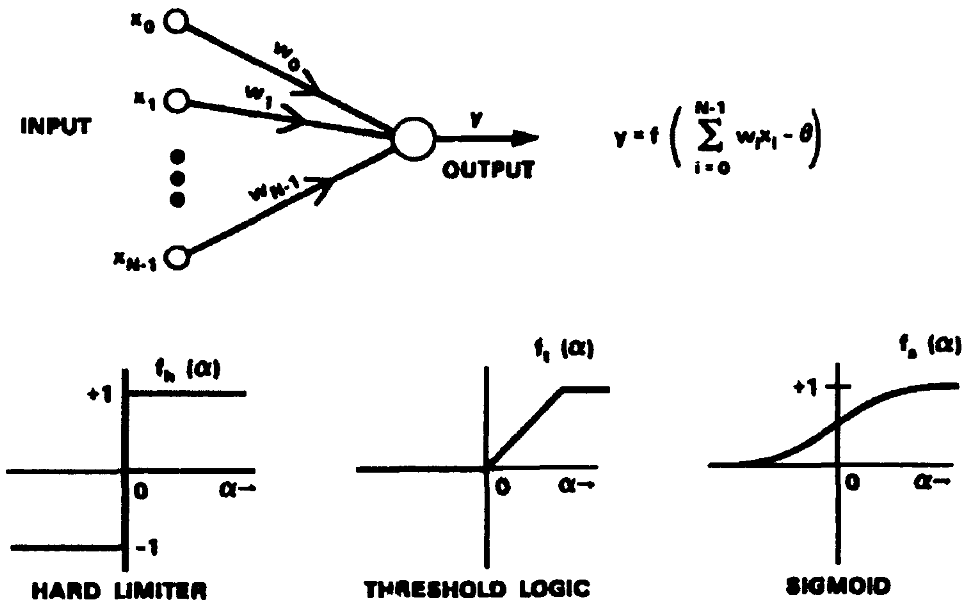


Figure 4. Types of Limiters

On the other hand, if the transfer function uses outputs of -1 and +1, the transfer sigmoid is defined as follows:

$$y = 1 - \frac{1}{1+x}$$

for $x \geq 0$ and

$$y = -1 + \frac{1}{1-x}$$

for $x \leq 0$ (Nelson, 1991: 47-48).

Regarding Nelson's two choices, Guyon states that a 0 is used instead of -1 in scaling for 'convenience' (Guyon, 1991: 220).

Regardless of the sigmoid function used, the objective is to allow the artificial neural network to emulate the brain. Within the brain, each nerve cell can respond to as many as 200,000 inputs, although 1,000 to 10,000 inputs are more typical. The inputs are either 'fire' or 'do not fire,' similar to a 0-1 variable (Nelson, 1991: 40).

These sigmoid functions provide the mathematical background for either firing or not firing.

2.2.6 Least Mean Square. The least mean square learning system (or LMS) is an alternative similar to the perceptron and serves as an introduction to the back propagation algorithm. In LMS, the output is now the real number which results from the equations as opposed to the adjusted integer used under the perceptron technique. Therefore, its output O is given as follows:

$$O = \sum_i w_i I_i + \theta$$

where w_i is the ith weight, I_i is the ith input, and θ is a constant or bias term. Notice that the LMS technique consists of one outpnt and one layer.

Under the LMS procedure, the true output, denoted as T, is an integer. The actual output, O, is real. The goal of LMS is to minimize the average square of the difference between these two values (Weiss, 1991: 87 - 88).

The training procedure under the perceptron procedure and LMS is the same for the most part. A key difference, however, is the adjustments. Since both T and O are limited to integers under the perceptron procedure, there is a chance for an exact match which would result in no adjustment. Under the LMS, on the other hand, exact matches are rare because O is a real number. Therefore, adjustments are more frequent under LMS (Weiss, 1991: 89).

LMS offers another advantage. Since LMS seeks to minimize the difference in terms of the minimum error distance, it tends to perform well even if the classes are not linearly separable (Weiss, 1991: 89).

LMS seeks to converge to a solution using the technique known as gradient descent. Gradient descent techniques have two weaknesses. First, they may oscillate and not converge. Second, if they do converge, they are not guaranteed to converge to the correct answer (Weiss, 1991: 89). To improve performance in spite of this picking of the local minima, Lippmann recommends the following alternatives: "allowing extra hidden units, lowering the gain term used to adapt weights, and making many training runs starting with different sets of random weights" (Lippmann, 1987: 18). To test for lack of convergence, Weiss recommends repeating the training sessions several times on the complete sample. If these different sessions, each starting from different initial weights selected at random, result in different final weights, a likely conclusion is that a local minima has been found. To correct for this local minima, Weiss recommends either decreasing the learning rate or increasing the momentum (Weiss, 1991: 106). The concept of learning rate and momentum arise in the back propagation algorithm, which will be covered next.

2.3 Back Propagation

The previous information (the multilayer perceptron, the training process, the sigmoid unit, and the least mean squares technique) come together in the back propagation algorithm. This back propagation algorithm is the key to this study of artificial neural networks.

Guyon praises the error back propagation algorithm as an "easy and elegant way of performing on-line (or *stochastic*) gradient descent to train neural networks" (Guyon, 1991: 226). The basic concept of this procedure is that outputs are first computed during the forward pass through a network of perceptrons. The output is next compared to the desired output. In the same manner, the discrepancy of the hidden unit outputs from the desired output level are computed during the back propagation pass and weights are adjusted accordingly. Finally, a least mean squares stopping criterion is used (Guyon, 1991: 227).

2.3.1 Back Propagation Algorithm. This section covers the actual back propagation algorithm. It is based primarily on the algorithms as presented by

Lippmann (Lippmann, 1987: 17) and Rogers (Rogers et al., 1990: 56). In general, it is a iterative sequence based upon minimizing the mean square error "between the actual output of a multilayer feed-forward perceptron and the desired output" (Lippmann, 1987: 17). For distinguishing between results, the back propagation algorithm uses the sigmoid function. The algorithm is a five step process.

Step One: Initialize the weights. Lippmann recommends initializing the weights to "small" random values (Lippmann, 1987: 17). Rogers states that the weights are normally from a uniform distribution between -.5 and .5 (Rogers et al., 1992: 56). Weiss recommends setting the initial weights randomly to numbers between 0 and 1 (Weiss, 1991: 85). The values chosen simply must not all be the same; if they are, the training will fail (Wiggins, 1991: 19).

Step Two: Feed data to the neural network to conduct the training process. The data contains the input values as well as the desired output. In a typical application such as pattern recognition, the desired pattern would be assigned an output of 1; all other outputs would be assigned a value of 0. Lippmann presents two alternatives for treating the training data set. First, all of the training data may be presented continuously. Second, the user may break up the training data into smaller sets, then use the different training data sets in a cyclical fashion. Repeat either of these two techniques until the weights stabilize. Lippmann does not recommend one technique over the other. This study will treat the training data as one complete set.

Step Three: Use the sigmoid nonlinearity to calculate the values of the outputs, a vector of ys.

Step Four: Adjust the weights as appropriate. This is where the back propagation algorithm gets its name because the algorithm starts at the output nodes and works backwards. Weights are adjusted according to the following equation:

$$w_{ij}(t+1) = w_{ij}(t) + \eta \delta_j x_i$$

where w_{ij} is the weight from node i (which may either be an input or a hidden node) to node j at time t; x_i is either the output of node i or is an input; η is a gain term or learning rate; and δ_j is an error term for node j.

The actual computation of δ_j depends upon where j is located in the artificial neural network. First, if j is an output node, then

$$\delta_j = y_j(1 - y_j)(d_j - y_j)$$

where d_j is the desired output of node j and y_j is the actual output.

On the other hand, if j is an internal hidden node, then δ_j is determined as follows:

$$\delta_j = x_j(1 - x_j)\sum_k \delta_k w_{jk}$$

where k is over all nodes in the layers above node j.

Lippmann concludes step four by noting "convergence is sometimes faster if a momentum term is added and weight changes are smoothed by" the following equations:

$$w_{ij}(t + 1) = w_{ij}(t) + \eta\delta_j x_i + \alpha(w_{ij}(t) - w_{ij}(t - 1))$$

where $0 < \alpha < 1$.

Step Five: Return to step two and repeat until either the maximum number of iterations is reached or the error is within tolerance.

2.3.2 Comments on Lippmann's Back Propagation Algorithm. Others have provided relevant inputs to the back propagation algorithm. Some of those comments will be covered next.

As with any gradient search technique, an important factor is knowing when to quit. Weiss recommends two possible stopping criteria which may be used in such a way that a stop is initiated as soon as either of the criteria is met. In addition to the previous technique of stopping after a predetermined number of epochs have been conducted, ANN training can stop after progress toward improving the minimization of the error distance is no longer being achieved. Care here must be taken to ensure a sufficient number of epochs occur between measurements since improvement on each individual epoch is not guaranteed under the back propagation procedure (Weiss, 1991: 101).

Under the Lippmann algorithm, a learning rate η and a momentum term α must be decided upon. Weiss recommends a learning rate of .5 and a momentum term of .9 (Weiss, 1991: 101). Rogers lists an example of using a learning rate of .3 and a momentum rate of .7 (Rogers et al., 1990: 56). According to Weiss, ideally, "one would like to use the largest learning rate that still converges to the minimum solution" (Weiss, 1991: 100). The goal of incorporating a momentum term is to speed the rate of convergence and avoid local minima (Weiss, 1991: 101). Finally, Wiggins recommends a learning rate of approximately .001 (Wiggins, 1991: 14).

Backward propagation is data hungry. Lippmann notes that "in many cases the number of presentations of training data required for convergence has been large (more than 100 passes through all the training data). Although a number of more complex adaptation algorithms have been proposed to speed convergence, it seems unlikely that the complex decision regions formed by multilayer perceptrons can be generated in few trials when class regions are disconnected" (Lippmann, 1987: 18).

Perhaps the most extensive comments on back propagation come from Weiss. He offers a similar algorithm for back propagation and notes that the back propagation algorithm possesses some strong points and some weak points. For example, even the slightest change in net structure may yield vastly different results. Weiss offers some alternatives for improvement. In the algorithm, the programmer has the option of altering the weights after going through the entire training set (after each epoch) or revising after each exemplar. Weiss recommends revising weights after each exemplar, "although revision by epoch may have the stronger theoretical foundation" (Weiss, 1991: 99). Hart concurs with this because it speeds up training, "especially in the early stages" (Hart, 1992: 218). Other suggestions for improvement include the following: presenting the training cases in random order; using biases randomly selected between -.5 and .5; normalizing inputs to numbers between 0 and 1; and repeating training sessions several times with different randomly selected initial weights (Weiss, 1991: 99 - 100).

Another potential weak point of the back propagation technique is the fact that, if enough hidden units are added to a network, one can eventually lower the error distance down to zero. However, the danger here is that of overfitting to the training data. Weiss gives the example that, with the addition of sufficient hidden units, one can get the error distance down to zero even if the sample data has an inherent Bayes error rate of 30% (Weiss, 1991: 103).

Weiss lists several potential problems with the back propagation procedures. He calls it a 'user's nightmare.' Those potential problems include the following: choosing the appropriate learning rate and momentum term, choosing to train by epoch or by pattern (case), finding an effective initial random starting state, and deciding when to stop. Changing any of these parameters may have a 'major' impact upon the results (Weiss, 1991: 108).

Weiss mentions a variation on the standard back propagation procedure which uses ordinary differential equations (ODE) and often yields superior results. This technique eliminates the need to adjust parameters and treats the differential equations as black boxes that can be called on without a detailed understanding of the mathematics occurring inside them (Weiss, 1991: 108).

2.4 Stopping Criteria

An important, yet relatively unexplored area of ANNs is determining when to stop ANN training. The goal of a stopping criteria should be to stop the learning process of an ANN before the ANN starts overfitting the exemplars (Ramamoorthy, 1989: 136).

Ramamoorthy and Shehkar discuss one method of determining a stopping point. They call for monitoring the classification error rate of a test set in addition to monitoring the classification error on the training set. Inherent in this is the assumption of the training and test sets being independent. As training occurs, the classification error should initially decrease for both sets. Ramamoorthy recommends watching the error of the test set as it decreases. When a general increase in the error classification curve of the test set occurs, it indicates that the training data set is being overfit and that training should cease. The error function has been minimized and this is the stopping point for the learning algorithm. This is referred to as a "transition from [an] underfitted model to overfitting of [the] model" (Ramamoorthy, 1989, 140).

Another criterion which may be monitored during ANN training is the error sum of squares. Using this criteria, the error is computed between the desired output and the ANN output. A key point to remember in using the back propagation algorithm is that the sigmoid function is not capable of providing perfect matches between the desired output and the ANN output. For example, regardless of the value of the input x, the output function

$$y = \frac{1}{1 + e^{-x}}$$

can not possibly give an output of 0 or 1, which is the desired response (Choie, 1991: 723). Therefore, when seeking to minimize the sum of squares for error, ESS, defined as

$$ESS = \frac{1}{2}\sum_i (y_i - d_i)^2$$

(where y_i is the computed output and d_i is the desired output), the sum can never reach 0.

When using ESS, the establishment of the critical value of ESS at which ANN training should stop is important. For example, assume the goal of ANN training is to train the network until the ESS is 0.04. In this case, every exemplar in the training set must be off by less than 0.2 from the desired response for training to terminate. Otherwise, any individual exemplar which is off by a value of 0.2 or higher unilaterally pushes the sum of function ESS to greater than 0.04 (because $0.2^2 = 0.04$) regardless of the error in all other exemplars. Therefore, a single exemplar may keep the algorithm from stopping if the stopping criteria is the function ESS (Choie, 1991: 724).

Because of these complications caused by one potential exemplar, Choie and others recommend a test which seeks the maximum error for each exemplar of some constant, say .5 (Choie, 1991: 724). They further point out that this stopping criteria, like the rest, has potential problems and should not be used universally. On the other hand, some note that squaring the error seems to lead to overlearning the training set and therefore recommend using the absolute error instead (Hergert, 1992: 980).

Another stopping criteria is proposed by Pan and Chen. They point out that a measure of completeness is the amount by which the weights are changing during each epoch. Therefore, they propose monitoring the following function:

$$E = \sum_{k=1}^{m}[w_k(t+1) - w_k(t)]^2$$

where m is the total number of weights and w_k is the weight of neuron k for the tth epoch (Pan and Chen, 1992: 360). Pan and Chen do not give a standard for how much weights can change before the optima is reached.

An extension of the Pan and Chen stopping criterion involves observing the average absolute relative weight change WC defined over n exemplars as follows:

$$WC = n^{-1} \frac{\sum_{k=1}^{n} |w_k(t+1) - w_k(t)|}{|w_k(t)|}$$

Using a third set of data, or a validation set, is another procedure. The validation data set is treated similar to the testing data set in that training ceases when the classification error rate on the validation set increases. Using a validation set in this way was recommended for the case where the training and test data sets contained noise (Hergert, 1992: 980). This thesis study will use the XOR problem and a mesh problem with known data; therefore the validation set will not contain noise. However, the principle of a third data set appears relevant even without considering the addition of noise.

In summary, the literature provides the following different suggestions on functions of the artificial neural network to track when deciding when to terminate ANN training.

- Total absolute error.
- The error sum of squares.
- The largest absolute error.
- Classification error rate for the training set or the testing set.
- Size of the change in weights between epochs.

Additionally, the following stopping criteria will be evaluated.

- A combination of the classification error rate for the training set, the testing set, and the validation set.
- A technique where each potential stopping criterion is a switch, stopping when a sufficient number of switches are turned 'on'.

III. Methodology

This chapter covers the decisions and techniques to be used in this evaluation. Specifically, it covers the following areas: introductory decisions, the problem to be solved, the FORTRAN program, potential stopping criteria to be considered, and evaluation criteria.

3.1 Introductory Decisions

3.1.1 Strategic Decisions. In beginning the project development, some strategic decisions were made regarding performing back propagation using multilayer artificial neural networks. Those decisions are detailed below.

- Selecting the data. Choices here included using actual or computer generated data. Since it was desired to know whether ANN learning had been successful, a great deal of data was needed. Therefore, the study used random, computer generated data.
- Selecting a data philosophy. Generally, there were two ways to treat the training versus the testing data. First, they could both have been in a larger data set from which the training and testing exemplars were drawn. Alternatively, the study could have used distinct training and testing sets. This study examined both approaches, keeping the training and testing exemplars separate in one part of the initial experiment and randomly assigning exemplars to the training and testing sets during the second part of the initial experiment in an attempt to prevent overlearning or overfitting the data in another part of the initial experiment.
- Evaluation philosophy. In the end, a technique must exist to decide which ANN solution was the 'best.' Alternatives already presented include the classification error rates on various data sets, the largest absolute error, the sum of squared

errors, and the speed of reaching a solution. The study used the classification error rate on a separate validation set. The validation set used was larger than the training and testing sets and is considered to represent the true population.

- Selecting the problems to solve. The study used two problems which are common in the study of ANNs: the Exclusive OR (XOR) problem and a meshed data set problem. These problems were chosen because of their relative simplicity and the large availability of data points.

3.1.2 Tactical Decisions. In addition to the strategic decisions, running the ANN requires some tactical decisions.

- Number of nodes and hidden layers. Both of these factors can affect the training time and the ability of the artificial neural network (ANN). For the purposes of deciding when to stop training, however, the number of nodes and hidden layers must be held constant as a controlled variable. Therefore, the general practice in the literature of having one hidden layer was used. The decision of how many nodes to place on the hidden layer is more arbitrary. As an initial experimental set up, three nodes in the hidden layer were used. For the meshed data set problem, the number of hidden nodes was varied from four to twenty (inclusive for even numbers only).

- Number of training vectors to use. Foley recommended using three times as many training vectors per class as there are features. Additionally, he noted that if the total number of training vectors for a two-class problem is less than twice the number of features, the error rate on the training set will be very close to zero (Rogers et al., 1990: 61). Since the beginning experiment called for two features with one hidden layer, a training set containing 100 exemplars was used. This number safely exceeded Foley's recommendation.

- Setting the learning rate and momentum rates. The current literature does not provide a consensus for these attributes of the ANN. Therefore, a learning rate of 0.35 as used by those with experience in ANNs such as Rogers (Rogers, et al., 1990: 56) will be implemented. Momentum, on the other hand, was treated differently. Although it offers some advantages in an ANN, its primary attribute is that it allows training to be sped up. Since this study was primarily interested in deciding when to stop training as opposed to how fast the ANN can get to that stopping point, momentum was omitted from the ANN algorithm. Remember additionally that the back propagation algorithm is based upon a form of the steepest descent, which is widely known to converge slowly. Therefore, the back propagation algorithm was expected to be slow and the study was not primarily concerned with the speed of convergence.

- Normalizing the input features. The general consensus in the literature on this question leans toward normalizing the input features to values between -1 and 1. This philosophy was followed. Using the XOR problem, all input features took on values between -1 and 1. If different inputs are used later, they must be normalized outside of the program before input.
- Updating weights after each exemplar or each epoch. Here again, the literature allows either decision to be correct. This study used the technique of updating weights after each exemplar.
- Randomizing the data. ANN training used the technique where the order of the exemplars were randomized in the training set before ANN training. Although the effect of not doing so is unknown, the effect can not be deleterious in any case.

3.2 Problem Selection

This study examines two problem data sets which will be descr'bed next.

3.2.1 The Exclusive OR Problem. An investigation of stopping criteria required an abundance of data where the correct output is known. Therefore, the Exclusive OR (XOR) problem was used for the pilot study.

The XOR problem is popular because of its relative simplicity and the ability to generate an abundance of data. It is a more complex region than can be handled by the single perceptron and the two regions can not be separated by a single hyperplane. Essentially, the data are divided as shown in Figure 5. The individual data points

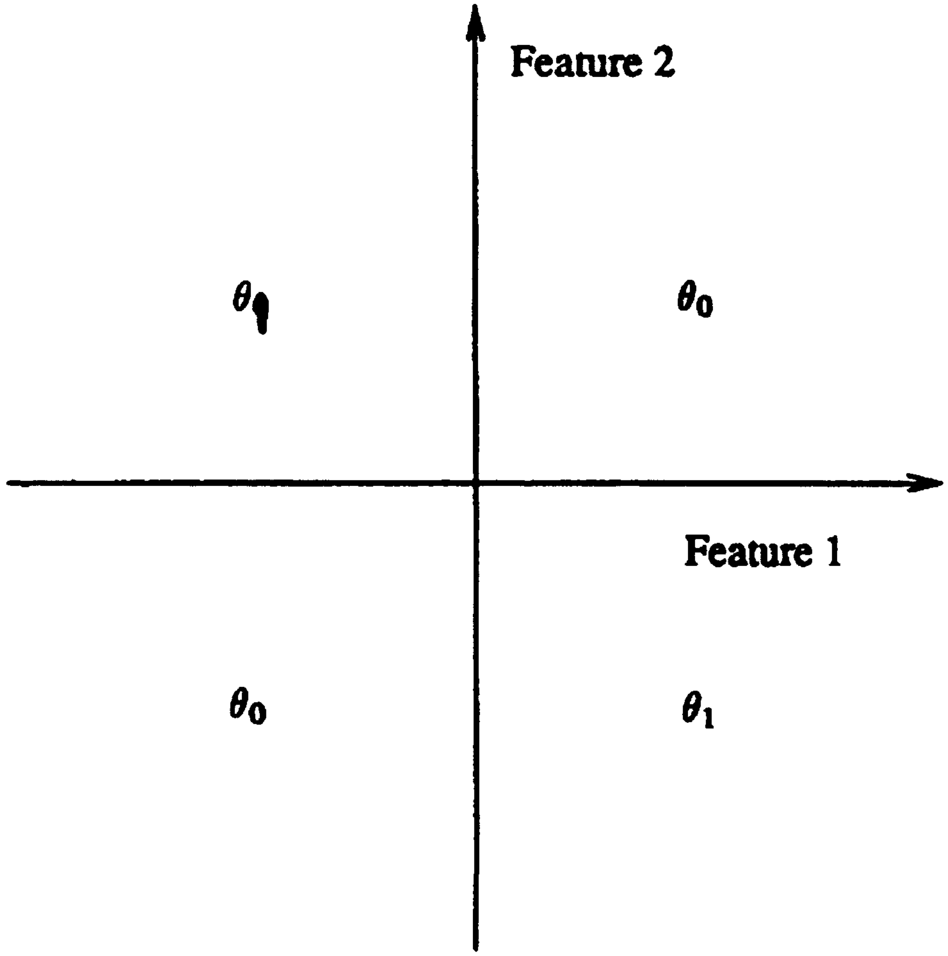


Figure 5. The XOR Problem

are divided based upon the quadrant in which they reside. For example, points in the area θ_0 are assigned a response of 0 and points in areas θ_1 are assigned a response of 1. Since any point within any of the regions can be used as an exemplar, an infinite number of data points exist for this problem (Rogers et al., 1990: 53).

Since the study used a hypothetical problem for research purposes, ensuring that the proper features were selected was not a concern. The possibility of deleting a relevant feature did not exist unless a random noise variable was added. Likewise, the goal of the thesis was to measure the relative attributes of different stopping criteria and not to find a response surface methodology of the various factors which may affect the learning rate. Therefore, the number of hidden nodes was held constant at three and the learning rate held constant at 0.35.

3.2.2 The Mesh Problem. The second problem examined will be referred to as a mesh problem because it contains a meshed data set. The mesh problem is more complicated than the XOR problem because it involves two regions which can not be separated by two lines. The solution space for each of the outputs for the mesh problem is more complicated and can best be described by a picture. See Figure 6 for a diagram of the two response spaces. Response 1 is the shaded area and the remainder of the space is assigned response 0.

3.3 The FORTRAN Program

The FORTRAN program begins by taking the XOR data and reading each exemplar into the appropriate array for the three individual data sets: the training set, the testing set, or the validation set. Each of these sets is used differently for training and evaluation purposes. The training set is used in training the ANN. The testing set is used primarily for gathering data for the stopping criteria. The validation set is used to represent the population when evaluating the goodness of the proposed solution.

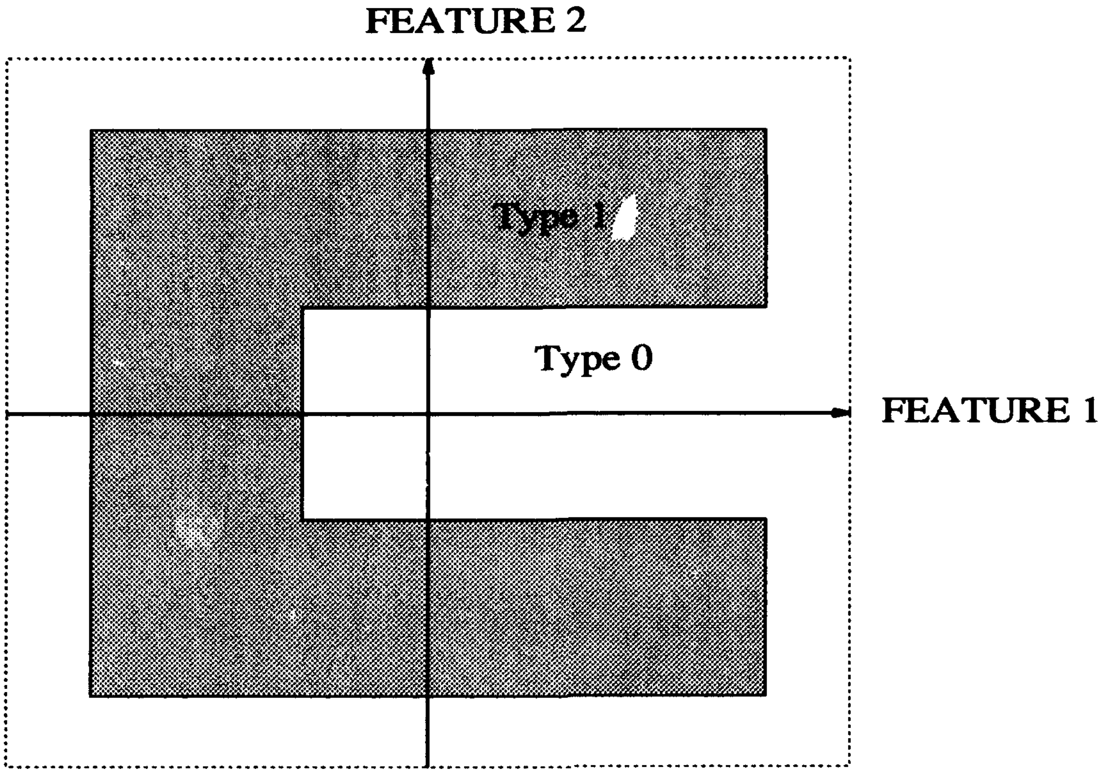


Figure 6. The Mesh Problem

After the data is assigned to the various sets, the ANN begins the learning process. After each epoch, each potential stopping criterion is queried to determine its current level and a new random order of training exemplars is assigned. The process continues for each set of beginning random weights until all stopping criteria have been tracked through the maximum number of epochs. Finally, the program checked to see if a sufficient number of replications of beginning weights have been used. At each replication, new initial random weights are assigned.

When enough replications of the program have been run, the output of each potential stopping criterion is compared to the classification error rate for the validation set as a representation of the output of the ANN on the population. An ideal predictor of when to stop ANN training should cross some threshold or exhibit some telltale characteristic at the point at which the classification error rate of the validation set is at a minimum. For a flow chart description of the FORTRAN program, see Figure 7.

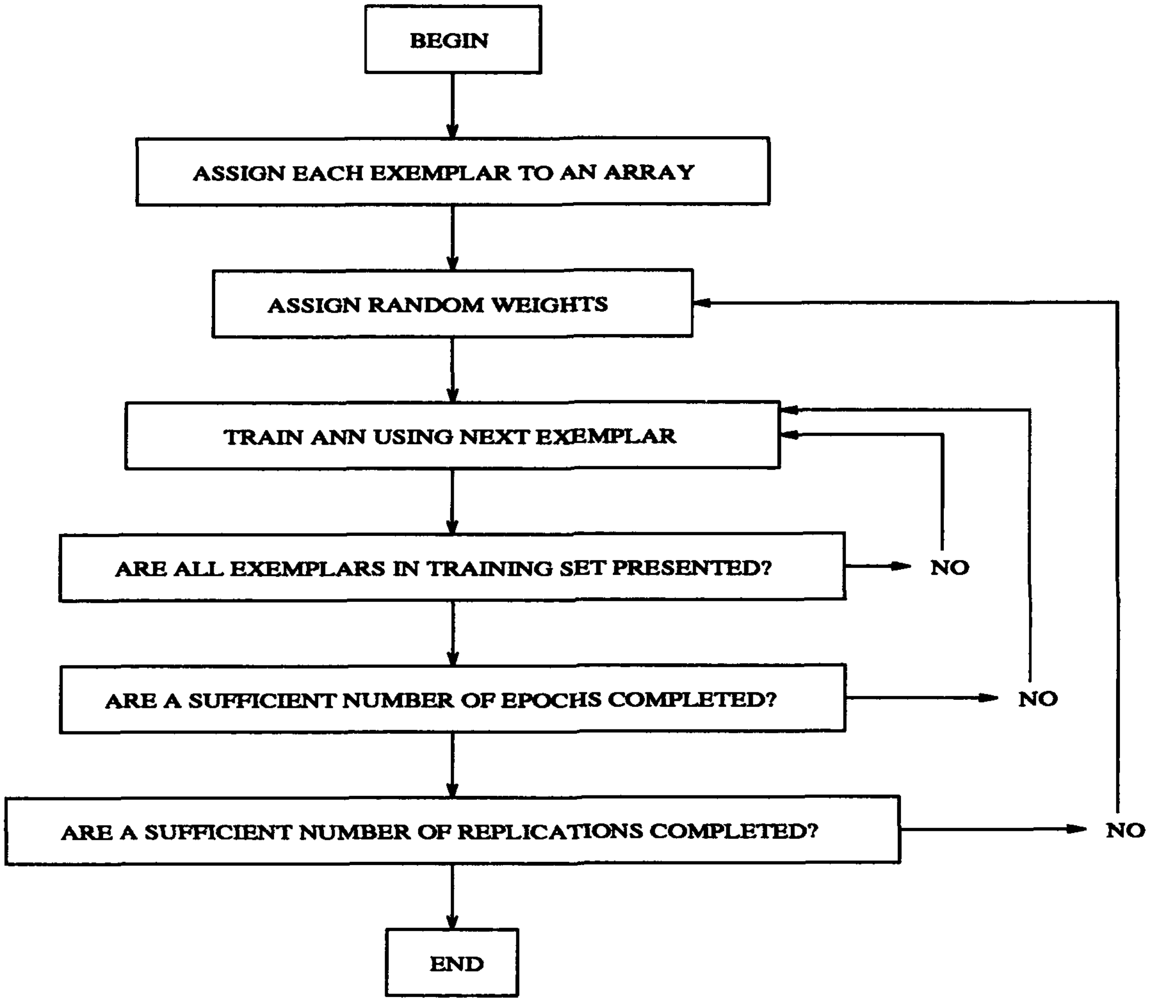

BEGIN
ASSIGN EACH EXEMPLAR TO AN ARRAY
ASSIGN RANDOM WEIGHTS
TRAIN ANN USING NEXT EXEMPLAR
ARE ALL EXEMPLARS IN TRAINING SET PRESENTED?
NO
ARE A SUFFICIENT NUMBER OF EPOCHS COMPLETED?
NO
ARE A SUFFICIENT NUMBER OF REPLICATIONS COMPLETED?
NO
END

Figure 7. FORTRAN Program Flow Chart

3.4 Stopping Criteria

The essential goal here was to stop training at the optimum point such that the ANN avoided overlearning and reached an acceptable error rate as soon as possible.

Therefore, the study must answer the question of how many times the training data should be presented.

Through reviewing the literature and by considering additional potential areas of study, the following potential stopping criteria were considered as part of this study:

- Total absolute error on the testing set.
- The error sum of squares on the testing set.
- The largest absolute error on the testing set.
- Classification error rate for the training set or the testing set.
- Size of the change in weights between epochs measured specifically in the following four ways: absolute weight change, squared weight change, relative weight change, and mean weight change.
- A combination of the classification error rate for the training set and the testing set as measured by the weighted average error rate.
- A measure where each potential stopping criteria is a switch, stopping when a sufficient number of switches are turned 'on'.

For the mathematical representation of these stopping criteria, please see Appendix B.

3.5 Evaluation Criteria

Should the FORTRAN program indicate potential stopping criteria, a method must exist to determine which stopping criterion is best. On this point, the literature was limited. Therefore, the following strategy was decided upon.

1. Choose the method which resulted in the lowest classification error rate on the validation set.

2. Choose the fastest convergence to a final set of weights. In other words, if all other things were equal, faster was better.

IV. Results

4.1 Introduction

This chapter covers experiments conducted using the artificial neural network (ANN) described in Chapter III. Chapter IV uses a chronological sequence of the experiments to lead the reader to the conclusion that the classification error rate, the moving average of the classification error rate, and the total absolute error (all computed using the testing set) are the best indicators of the proper time to terminate ANN training. However, the initial ANN settings must first be enumerated to put the results in the proper context.

4.2 Initial Settings

The initial version of the ANN was trained on the Exclusive OR (XOR) problem using the parameters which follow. It was trained using both the case where the training and testing exemplars were kept separate throughout the experiment and the case where each exemplar was randomly redesignated as belonging to either the training set or the testing set at the beginning of each epoch. Parameters were set as follows:

1. Maximum number of iterations: 1500.
2. Weighted Average Ratio of (training set to testing set): $\frac{.5}{.5}$ or 1 to 1.
3. Learning rate (η or eta): fixed at 0.35
4. Moving average interval: 5 most recent observations.
5. Size of the training set: 100 exemplars.
6. Size of the testing set: 100 exemplars.
7. Size of the validation set: 600 exemplars.

4.3 Initial Experiment

The goal of the pilot study was to monitor the status of the potential stopping criteria to determine whether any of them change in a manner similar to the classification error rate of the validation set, which was used to represent the true population. Due to data limitations occurring in the real world, the actual classification error rate on the validation set (the population) would be unknown to the ANN programmer. The classification error rate for the validation set recorded during one replication is given in Figure 8. If a potential stopping criterion reaches a threshold or exhibits a measurable change at the point at which the validation set classification error rate is at a minimum, then it seems reasonable to believe that this stopping criterion may represent an indicator of the proper time to terminate ANN training.

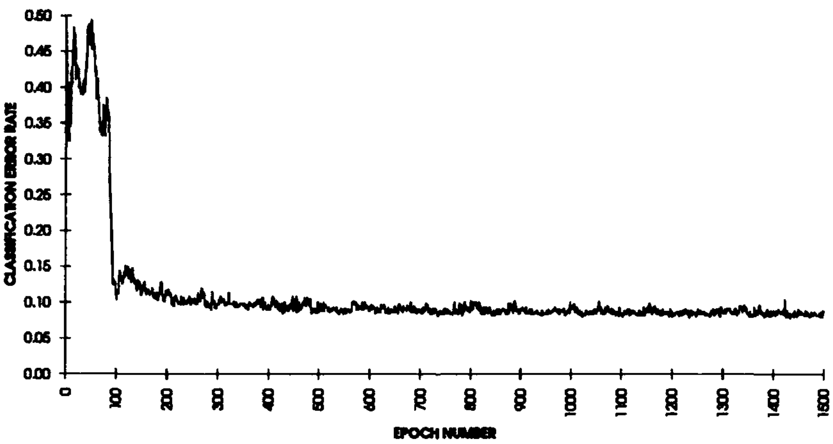


Figure 8. Classification Error Rate on Validation Set

In the initial experiment, the following items were tracked as potential stopping criteria. Mathematical formulations of these criteria are given in Appendix B.

1. Classification error rate on the training set.
2. Classification error rate on the testing set.
3. Classification error rate on a weighted average of the training and testing sets.
4. Moving average classification error rate on the testing set.
5. Total absolute error on the testing set.
6. Error sum of squares on the testing set.
7. Largest absolute error on the testing set.
8. Absolute weight change.
9. Relative weight change.
10. Squared weight change.
11. Mean weight change.

4.3.1 Failed Criteria. Based upon initial runs of the program using the XOR problem, the criteria which follow were judged as relatively weak indicators of the proper time to terminate ANN training and were not selected for further study consideration.

4.3.1.1 Error Sum of Squares. The error sum of squares (on the testing set) of the difference between the ANN output and the desired response yielded decreasing values throughout ANN training. Figure 9 shows the relationship of the error sum of squares value to the epoch in which it was received. However, the resulting differences in error sum of squares values were difficult to evaluate. The difficulty arose from the fact that all of the differences between the desired outputs and the ANN outputs were less than one. Squaring this difference yielded

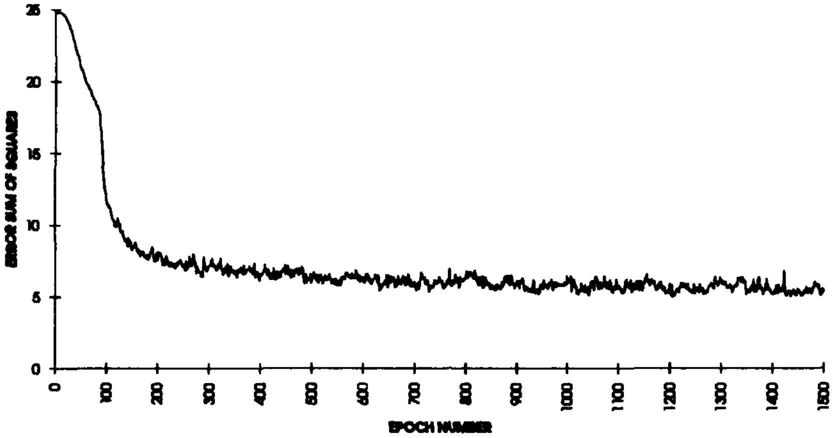


Figure 9. Error Sum of Squares

an even smaller value. While summing these values provided insight as to the status of training, it used smaller (and therefore harder to discriminate) values than the total absolute error stopping criterion which will be discussed later. The error sum of squares was not studied any further because the total absolute error stopping criterion yielded similar, more easily distinguishable results using essentially the same information.

4.3.1.2 Largest Absolute Error. Upon selecting the largest absolute error on the testing set as a potential stopping criterion, it was believed that, as ANN training occurred, the largest absolute error would decrease. While a largest absolute error of less than 0.5 would indicate a 0.0% classification error rate and therefore was an unrealistic standard, expecting the largest absolute error to decrease seemed logical. If the largest absolute error was selected as an indicator for stopping

ANN training, training would be terminated when the largest absolute error got sufficiently small. However, the experiment revealed that, as training occurred and the classification error rate decreased, the largest absolute error increased to values approaching 0.9 as seen in Figure 10. Off line, a slight program modification was

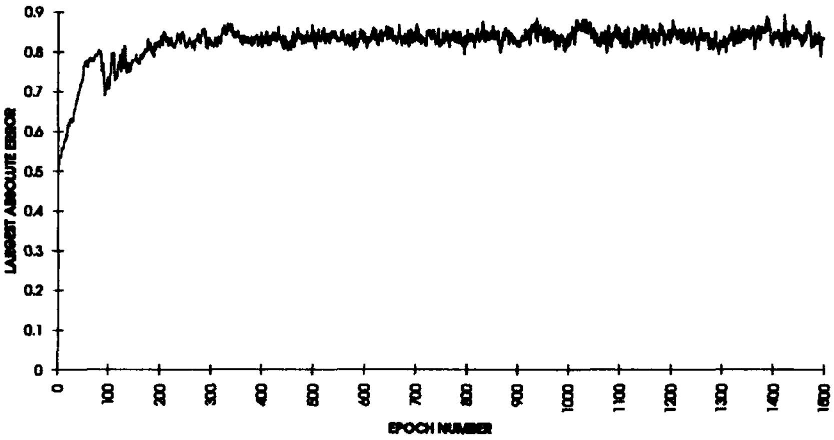


Figure 10. Largest Absolute Error

made to investigate why this occurred. In the modification, the exemplar which yielded the largest absolute error was recorded for each epoch. In nine cases out of ten, the same exemplar ($x_1 = -0.0515647, x_2 = 0.239364$) resulted in the largest absolute error from at least epoch 155 until the end. On average, this occurred sooner than epoch 155. In the tenth case, exemplar $x_1 = -0.883986, x_2 = 0.0143936$ caused the largest absolute error from epoch 44 until the maximum of 1500 epochs was reached. In summary, the ANN appeared to have picked out one exemplar close to an axis (the classification border) and gravitated toward that exemplar as far as the largest absolute error was concerned. Waiting for one exemplar to be sufficiently incorrect did not seem like a logical reason to terminate a training program which was designed to find the correct answer. Therefore, the largest absolute error was eliminated as a potential stopping criterion.

4.3.1.3 Weighted Average Error Rate. The weighted average error rate was designed in an attempt to compensate for the tendency of some ANNs to overlearn. Overlearning can be detected when the classification error rate on the testing set begins to increase while the classification error rate of the training set continues to decrease. By using part of the training set and part of the testing set, it was expected that an optimal stopping point would occur somewhere around the point at which the increase in the classification error rate of the testing set occurred. However, this expected increase in the classification error rate for the testing set never occurred. Refer to Figure 11. Therefore, for the particular XOR problem used here, using a weighted average error rate provided no additional benefit for this problem and was eliminated as a potential stopping criterion.

4.3.1.4 Absolute Weight Change. The weight change between epochs was monitored because a large weight change would seem to indicate large changes in the ANN function and absence of a local optima. Conversely, a small weight change would seem to indicate a relatively stable function in the current location. However, the resulting absolute weight change between epochs yielded results too inconsistent to be used for determining when to stop training. It was expected the absolute weight change would decrease to a lower level around the ANN optima. However, the absolute weight change varied throughout training in what appeared to be a manner independent of the error classification rate of the validation set. Refer to Figure 12. Therefore, absolute weight change was eliminated as a potential stopping criterion.

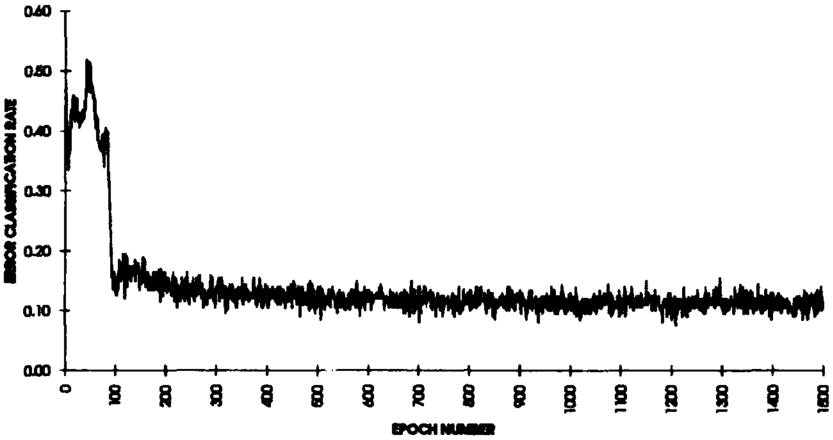


Figure 11. Weighted Average Classification Error Rate

4.3.1.5 Squared Weight Change and Mean Weight Change. The squared weight change and mean weight change were essentially different ways of computing the absolute weight change. Likewise, they yielded inconsistent results and were eliminated as a potential stopping criteria.

4.3.2 Remaining Criteria. After the above potential stopping criteria were eliminated, two criteria remained: the total absolute error on the testing set and the relative weight change.

4.3.2.1 Total Absolute Error. The total absolute error on the testing set as measured by the sum of the absolute value of the difference between the desired response and the ANN output indicated a potential reason for determining when to stop ANN training. The total absolute error (see Figure 13) decreased rapidly early

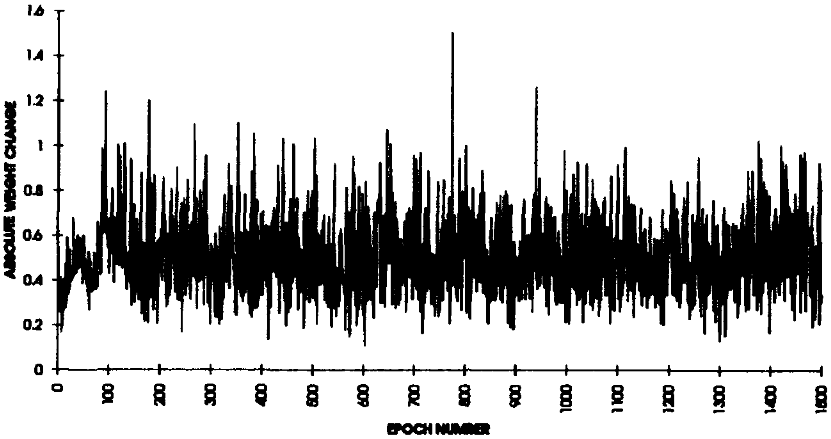


Figure 12. Absolute Weight Change

in training and leveled off at approximately the same epoch as the classification error rate of the validation set leveled off. This mirroring of the classification error rate on the validation set indicated that the total absolute error might be a good indicator of the proper time to terminate training.

4.3.2.2 Relative Weight Change. The relative weight change remained a valid criterion to be tracked, although it was assumed at the time that it showed potential only as an eliminating factor. If the relative weights were changing rapidly, it would seem to indicate that the ANN was not near a local optima, that the function was relatively unstable and changing rapidly, and that ANN training should continue regardless of the levels of the other potential stopping criteria. If, on the other hand, the relative weight change was sufficiently small, ANN training would be allowed to terminate should other factors indicate that a local optima was

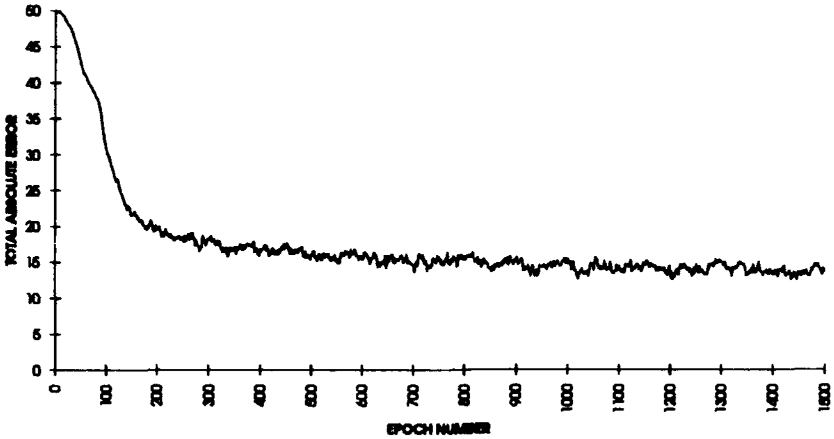


Figure 13. Total Absolute Error

near. Figure 14 shows relative weight change throughout training. Changes or spikes much larger than those indicated here were not uncommon in other cycles of ANN training. Given the high frequency of low relative weight changes early in ANN training, the usefulness of relative weight change as a single indicator of the proper time to terminate ANN training appeared unlikely. However, relative weight change was still tracked due to insufficient evidence to eliminate it completely as a potential stopping criterion at this point.

4.3.2.3 Previously Used Criteria. Training and testing set classification error rates remained valid indicators of the progress when training an artificial neural network. However, classification error rate on the training and testing sets had been previously evaluated as a method of determining when to terminate ANN training. The goal of this study was to investigate new, different reasons to stop

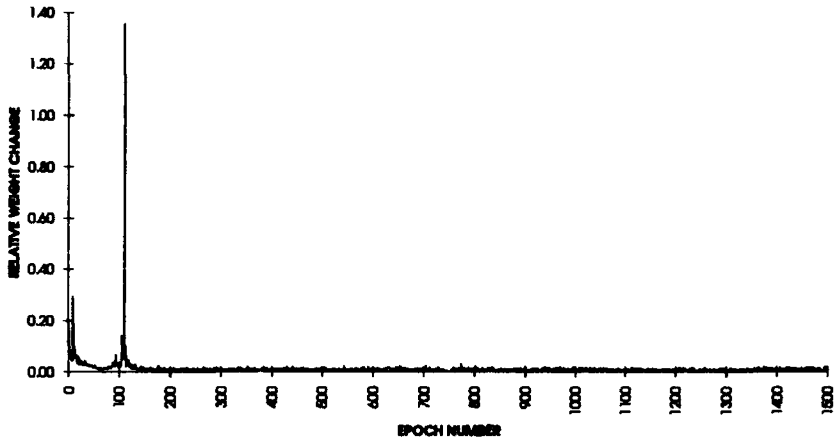


Figure 14. Relative Weight Change

training. Therefore, the study primarily looked at other measures for determining when to stop ANN training.

The moving average error rate is basically a smoother version of the classification error rate of the set (either training or testing) against which it is measured. Therefore, the same logic was applied to the moving average error rate as was applied to the training and testing sets; that is, they have been previously evaluated. All of these previously evaluated criteria remained as something which were tracked, however.

4.3.3 Initial Experiment Summation. Therefore, the following stopping criteria remained valid for continued consideration in the search for new criteria for determining the proper epoch at which to terminate ANN training.

1. Total absolute error.
2. Relative weight change.

In summing up the initial experiment, two additional areas must be covered. First, recall that the maximum epoch setting for the initial experiment was 1500 epochs. By observing the classification error rate of the validation set over ten cycles of 1500 epochs each, it was observed that no significant improvement occurred after 600 epochs (see Figure 8). Therefore, a maximum of 600 epochs was used for future experiments given the current values of η (or learning rate) and the current number of ANN hidden nodes and layers.

Finally, recall that the initial experiment included an evaluation of the differences between using separate training and testing sets and randomly assigning an exemplar to either the training or testing set at the beginning of each epoch. This paper will discuss this issue by using the total absolute error results, which were typical of the output received in all of the other potential stopping criteria categories.

When deciding when to stop ANN training, a consistent output from a potential stopping criterion is needed for that criterion to yield a consistent point at which it indicated a time to stop training. The random assignment of exemplars failed to meet this requirement. Outputs from the measured characteristics varied widely using random exemplar assignment. For example, while a total absolute error of between fifteen and twenty appeared to coincide with the occurrence of a minimum classification error rate for the validation set in every case when the training and testing sets were kept separate, the total absolute error never reached this threshold in some of the cases where the exemplars were randomly assigned at the beginning of each epoch. If a standard were set using the random assignment case, it would not be reached consistently. Similar results were received for other potential stopping criteria. Therefore, random exemplar assignment was eliminated and future experiments included only the case where the training and testing sets were kept separate throughout training. Recall, however, that this was expected. The random exemplar assignment technique was designed in an attempt to compensate for overlearning. Since overlearning was not observed and the above inconsistent function outputs did occur, it was decided to discontinue the practice of random exemplar assignment. The practice of random exemplar assignment may still be useful should overlearning occur.

4.4 Second Experiment

Using the output from the first experiment and comparing the classification error rate of the validation set with the total absolute error and the relative weight change, certain tendencies of ANN stopping criteria were noted. First, the classification error rate on the validation set appeared to reach a minimum around the time that the total absolute error was approximately 20. A total absolute error of 20 in this case equates to an average absolute error of 0.2 because 100 exemplars were used in the training set. The second tendency noted was that relative weight changes in excess of five percent appeared significant. Here, significant meant that the ANN function did not appear to be stable in areas where the relative weight change exceeded five percent. Therefore, an experiment was designed around these points to test differing levels of these two potential stopping criteria on the ANN responses. Specifically, an experiment was designed to record the classification error rate on the validation set and the speed of reaching a solution (as measured by the epoch at which a simulated solution was selected) at varied levels of the two remaining potential stopping criteria. Results of this experiment follow in two parts.

4.4.1 Single Stopping Criterion. For the first half of the second experiment, each of the two remaining potential stopping criterion were treated individually. That is, when the level of either reached a given threshold for that potential stopping criteria, it would serve as a signal to terminate ANN training for that criterion and threshold level only. At this termination point, the classification error rate on the validation set and the current epoch were recorded. Using three levels for each potential stopping criterion, nine total results would be explored.

4.4.1.1 Total Absolute Error. Under the single stopping criterion portion of the experiment, ANN training was terminated when the total absolute error reached the threshold levels of 15, 20, and 25. Total absolute error was defined as the sum of absolute values of the difference between the ANN output and the desired output. Results can be found in Figure 15, Figure 16, and Table 1. Both the classification error rate on the validation set and the total absolute error decreased as training occurred. These results were expected and they appeared consistent with Figure 8 and Figure 13.

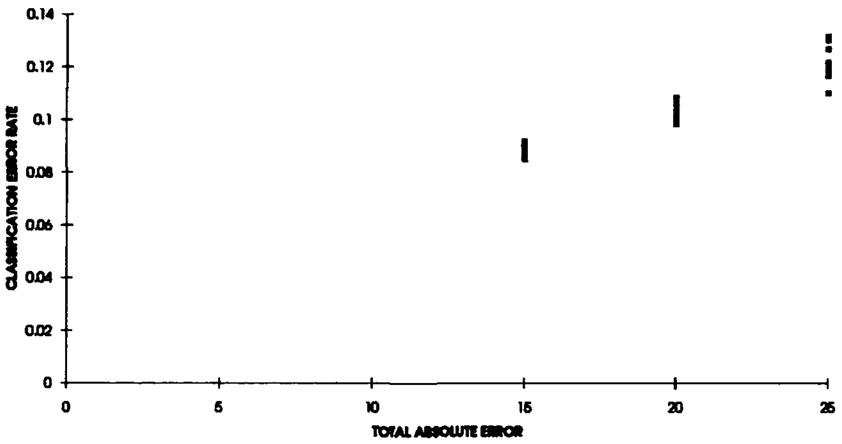


Figure 15. Second Experiment: Classification Error Rate Versus Total Absolute Error

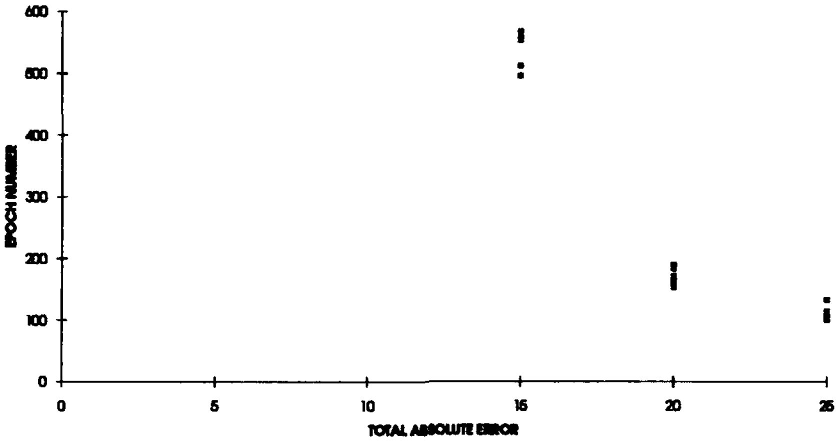


Figure 16. Second Experiment: Completion Epoch Versus Total Absolute Error

These results indicated that as the number of epochs increased, total absolute error on the testing set decreased and the classification error rate on the validation set decreased. This appeared to be consistent with the theory that the total absolute error might be an indicator of when to terminate ANN training. A key remaining question was whether total absolute error was an indicator of classification error rate, and therefore a reason to terminate training, or whether the two characteristics just happened to move together. The possibility that they may have just moved together may be seen in Figure 13, where the total absolute error continued to decrease slightly even after epoch 600, while Figure 8 showed that no significant change in the classification error rate occurred after epoch 600.

4.4.1.2 Relative Weight Change. In this section of the experiment, ANN training was terminated when the relative weight change reached the threshold

Threshold	Number of Occurrences	Mean Epoch	Epoch Standard Deviation	Mean Error Rate	Error Rate Standard Deviation
15.0	6	542	30	0.0886	0.002
20.0	9	170	14	0.1033	0.003
25.0	9	109	11	0.1230	0.007

Table 1. Second Experiment: Total Absolute Error

levels of 0.01, 0.05, and 0.09. Results can be found in Figure 17, Figure 18, and Table 2.

Threshold	Number of Occurrences	Mean Epoch	Epoch Standard Deviation	Mean Error Rate	Error Rate Standard Deviation
0.01	10	105	10	0.1780	0.110
0.05	10	12	14	0.4123	0.057
0.09	10	7	6	0.4340	0.028

Table 2. Second Experiment: Relative Weight Change

These results indicated that, as the relative weight change threshold decreased, the number of epochs required to reach a solution increased and the classification error rate decreased. Overall, this seemed to indicate that the relative weight change should remain valid as an indicator of when to terminate ANN training. Relative weight change at the lowest level (less than 0.01) has a classification error rate of nearly 18%, but the chance of an outlier was relatively large, with two appearing in this case. Also note that the occurrence of the outliers caused the variance to increase as the mean classification error rate decreased.

Due to the relatively high classification error rate (when compared to that achieved by the total absolute error on the testing set criterion) and the relatively high variance of the classification error rate caused by the occurrence of outlier cases,

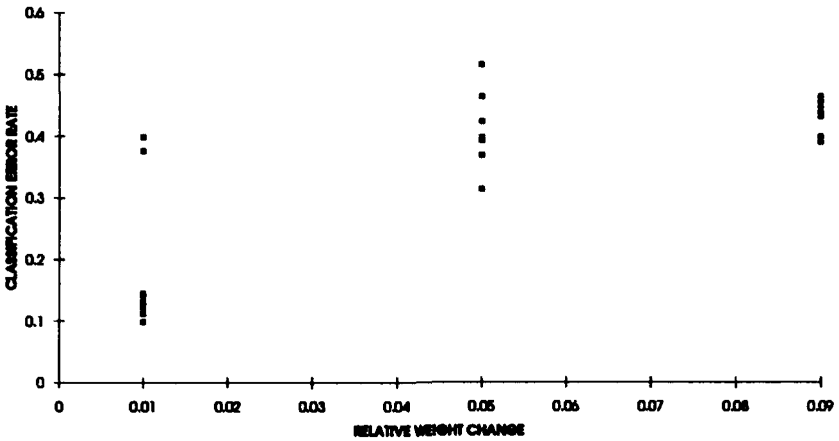


Figure 17. Second Experiment: Classification Error Rate Versus Relative Weight Change

a decision was made to continue to track the relative weight change, but only as an eliminating factor. By looking at the raw data, using the relative weight change as a primary determinant of stopping ANN training appeared unacceptable. Levels of relative weight change of less than one percent were rare and appeared to be mainly a hit or miss proposition which was searching for extremely small differences. Relative weight changes of less than one percent did occur, but they occurred randomly and were too inconsistent to indicate a proper epoch at which to terminate ANN training. Therefore, relative weight change remained under consideration only as an eliminating criterion.

4.4.2 Two Phase Stopping Criterion. To further determine the impacts of the relative weight change and the total absolute error as measurements of stop-

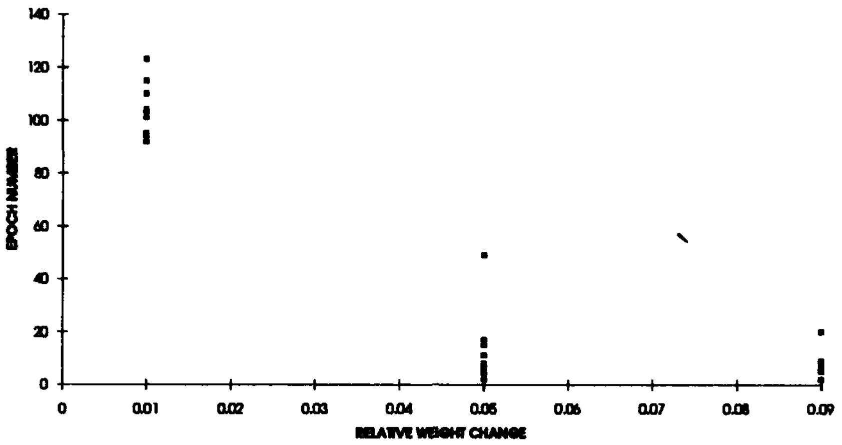


Figure 18. Second Experiment: Completion Epoch Versus Relative Weight Change

ping criteria, a two step stopping point criterion was implemented. Here, the total absolute error stopping criterion was varied from 15 to 25 in five uniform steps of 2.5 each and the relative weight change stopping criteria was varied from 0.01 to 0.09 in five uniform steps of 0.02 each. Both criteria had to be met before the ANN would recognize a requirement to terminate training. To evaluate these potential stopping criteria over the ten replications, the program measured a response of the mean classification error rate (see Figure 19) and the time to reach a recommended solution as determined by the mean current epoch at the time both thresholds were met (see Figure 20). The levels of total absolute error and relative weight change used here were selected heuristically from those lower levels occurring during ANN training. When comparing Figure 19 and Figure 20, note that both the axis for total absolute error and the axis for relative weight change were inverted between

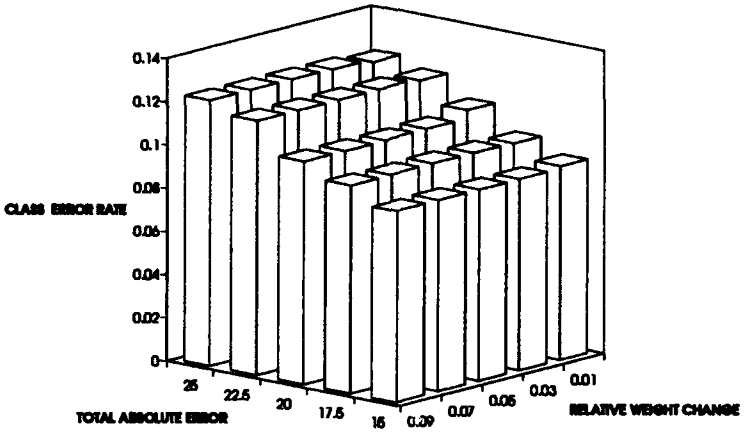


Figure 19. Second Experiment: Classification Error Rate Using Two Criteria

the two figures. This was done for presentation purposes to show bars which would otherwise be hidden in the rear of the figure.

Figure 19 and Figure 20 indicate the relative unimportance of the relative weight change criterion. As the relative weight change was varied in this experiment, the mean classification error rate and the mean completion epoch remained relatively stable. The main determinant of the classification error rate and the epoch of completion in this double stopping criteria experiment was the threshold level of the total absolute error.

4.5 Third Experiment

The second experiment showed that total absolute error was the only remaining factor which could be used as a determinant for stopping ANN training in this

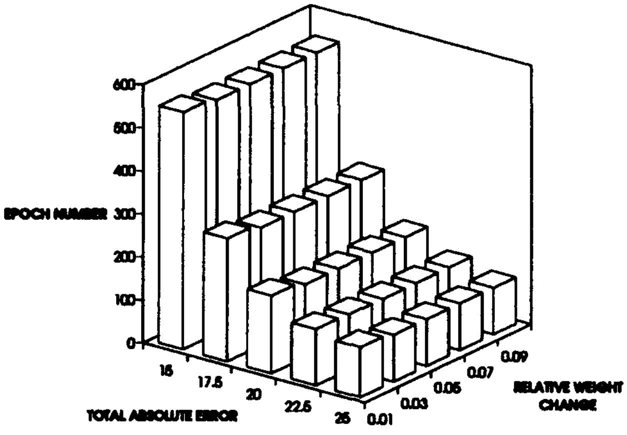


Figure 20. Second Experiment: Completion Epoch Using Two Criteria

particular network design. However, the limits of how low total absolute error would go within the 600 epoch limit established here were not yet known. The second experiment only tested total absolute error at five levels between 15 and 25. Therefore, a third experiment was designed to provide a closer examination of total absolute error. In particular, a program modification was made to find the resulting classification error rate and epoch of completion when the levels of stopping ANN training for the total absolute error were set at whole number increments of 0 to 20.

4.5.1 The Total Absolute Error Function. The results of this experiment indicated that, through chance, the level of fifteen which was selected for the second experiment was indeed the lower limit for total absolute error in this particular network design. The curve was decreasing, asymptotically approaching values just less than 15 (see Figure 21 and Figure 22). In fact, the total absolute error decreased

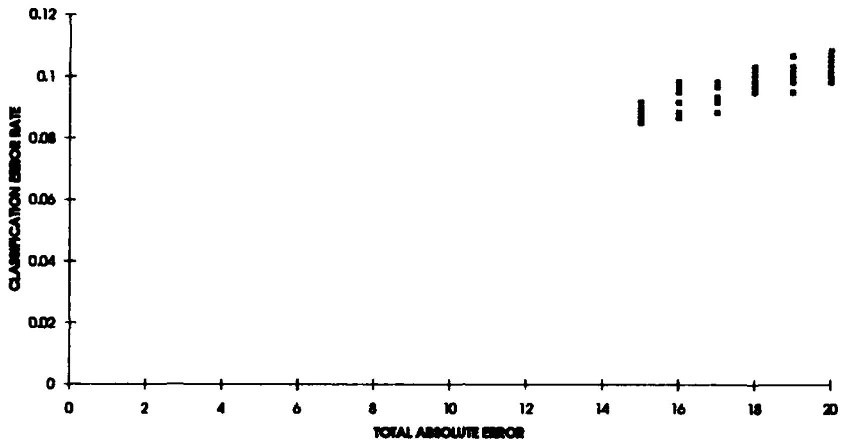


Figure 21. Third Experiment: Classification Error Rate Versus Total Absolute Error

to the level of 15 in only six of the ten test cases in this experiment. No observations occurred at the experimental levels of 14 or lower.

4.5.2 The Total Absolute Error Conclusions. Within the 600 epoch limitation resulting from the observation that the classification error rate for the population (the validation set in this case) did not decrease beyond that epoch, the lowest that total absolute error went for this particular experiment was 15. Table 3 shows the details of the total absolute function at the various levels throughout training. As the total absolute error decreased, the classification error rate and the variance of that classification error rate also both decreased. Therefore, a higher probability of selecting an ANN which had both a smaller true classification error rate and a

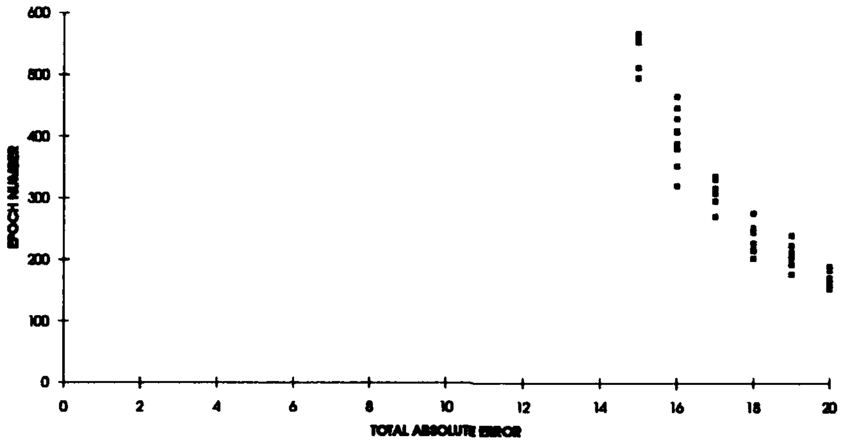


Figure 22. Third Experiment: Completion Epoch Versus Total Absolute Error

smaller variance of the classification error rate was achieved by selecting a random set of ANN weights based upon the lowest total absolute error of the testing set.

Perhaps as significant as noting the behavior of the total absolute error function when the threshold levels in Figure 21 were reached was noting how the total absolute error function reacted when the threshold levels were not reached. Notice in Table 3 that the threshold level was reached in nine out of ten replications. In the replication in which no threshold levels were reached, the total absolute error function remained greater than 33 throughout 600 epochs of training and the final classification error rate for the validation set was approximately 30%. The results indicated that the classification error rate on the validation set was not at the lowest levels if the total absolute error function was not at the lowest level. Figure 13 showed the total absolute error function for a replication which met the threshold level of 15. For

Threshold	Number of Occurrences	Mean Epoch	Epoch Standard Deviation	Mean Error Rate	Error Rate Standard Deviation
15	6	542	30	0.0886	0.002
16	9	400	45	0.0935	0.004
17	9	311	19	0.0937	0.003
18	9	238	22	0.0991	0.003
19	9	205	19	0.1004	0.003
20	9	170	14	0.1033	0.003

Table 3. Third Experiment: Total Absolute Error

an examination of the total absolute error functions which did not meet all of the threshold levels of Table 3, see Appendix C.

4.6 Mesh Problem Experiment

4.6.1 Mesh Problem Set Up. A sequence of steps to this point had led to a solution for the XOR problem, but the question remained whether these same steps could assist in solving a different problem. As noted earlier, the mesh problem involved a more complicated space for the two classes. Therefore, the ANN was allowed to train for a greater number of epochs. After inconclusive results were reached using a network with a maximum number of epochs set at 1500, the ANN was allowed to train for 3000 epochs. An example of the classification error rate received during one replication is given in Figure 23. The classification error rate in Figure 23 is formed from an evenly weighted average of the classification error rate of the training and testing sets for one of the ten replications. A weighted average was used because, when attempting to use an artificial neural network to classify data, the classification of the population would be unknown. Therefore, the ANN was only given as much data as was considered known in the problem. The validation set data was treated as an unknown population.

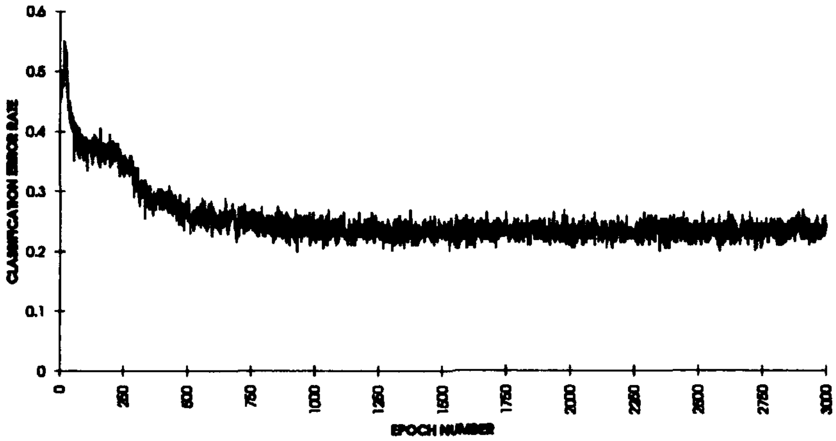


Figure 23. Mesh Problem Classification Error Rate

Figure 23 resulted in the assumption that no further learning occurs after epoch 2000. However, the ANN was allowed to train up through epoch 3000 for evaluation purposes.

4.6.2 Mesh Problem Results. Mesh problem results are given in a format similar to that used for the XOR problem. A comparison of the epochs required to obtain a solution and the classification error rate are given in Figure 24 and Figure 25 respectively. Table 4 contains the average data results.

Figure 24 and Table 4 illustrate a key finding regarding the total absolute error on the testing set function. When the lowest total absolute error occurred (at a value of approximately thirty). the average classification error rate on the validation set was greater than the average classification error rate recorded at a total absolute error value of approximately thirty-one. This demonstrated that simply terminating

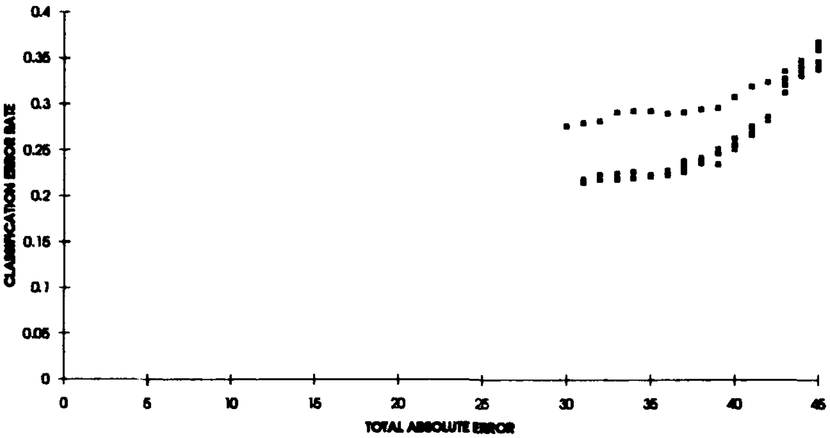


Figure 24. Mesh Problem: Classification Error Rate Versus Total Absolute Error

ANN training at the lowest recorded value of the total absolute error on the testing set did not guarantee that the ANN would yield the lowest classification error rate on the validation set.

Using the total absolute error technique of deciding upon the point at which to stop ANN training resulted in a higher classification rate for the mesh problem than for the XOR problem. Given the more complicated class divisions of the mesh problem, this was not unexpected. Figure 26 gives the solution space of the ANN output to visually quantify the mesh output results. The classification error rate of this particular network design never decreased below 20%. In fact, in only three of the ten replications did the classification error rate on the validation set decrease below 30%. The classification error rate of the ANN design seen in Figure 26 was 22.83% and was achieved after 50,000 epochs.

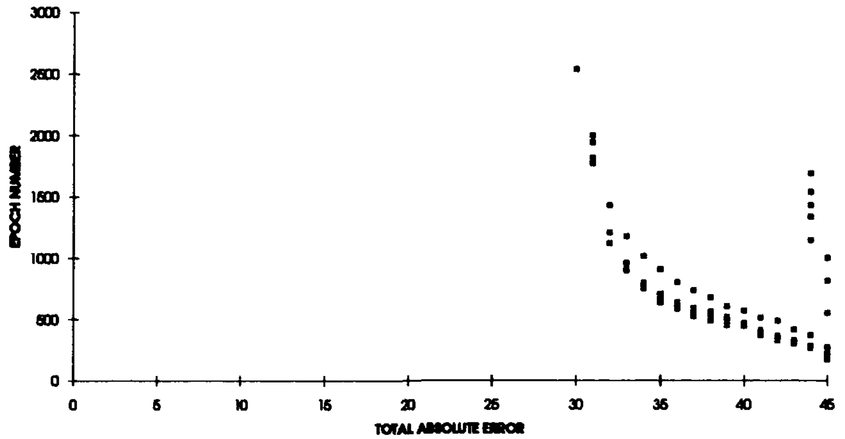


Figure 25. Mesh Problem: Completion Epoch Versus Total Absolute Error

Since the other replications yielded similar solution spaces, it appeared that an ANN with three hidden nodes was insufficient to map the many corners found in this particular mesh design. Neither the total absolute error nor the classic techniques (such as stopping training at the lowest level of the testing set classification error rate) mapped the solution closely. All stopping criteria techniques failed to detect the corners of the meshed data set. The ANN proposed solution seemed to encircle the largest mass as opposed to attempting to search out the corners of the solution space.

Therefore, it was decided to test the ANN with a different number of hidden nodes to see if more hidden nodes would allow the ANN to 'get around the corners' of the mesh problem. An ANN with eight hidden nodes was trained for up to 15,000 epochs. A graph of the ANN output resulting from this experiment is given in

Threshold	Number of Occurrences	Mean Epoch	Epoch Standard Deviation	Mean Error Rate	Error Rate Standard Deviation
30.0	1	2535	N/A	0.2767	N/A
31.0	5	1891	96	0.2290	0.029
32.0	5	1232	115	0.2337	0.027
33.0	5	964	121	0.2360	0.031
34.0	5	826	107	0.2383	0.031
35.0	5	715	108	0.2367	0.032
36.0	5	650	88	0.2387	0.029
37.0	5	590	84	0.2437	0.027
38.0	5	554	74	0.2500	0.025
39.0	5	511	59	0.2567	0.024
40.0	5	466	59	0.2673	0.023
41.0	5	415	57	0.2813	0.022
42.0	5	376	62	0.2927	0.018
43.0	5	334	45	0.3227	0.010
44.0	10	861	611	0.3410	0.006
45.0	10	390	294	0.3572	0.010

Table 4. Mesh Problem: Total Absolute Error

Figure 27. This solution space more closely maps the true solution space given earlier in Figure 6. Under this revised network, the classification error rate on the validation set was able to decrease to less than 30% in all ten replications and obtained levels of less than 20% in seven of the ten replications. The mesh solution space graph given in Figure 27 resulted in a classification error rate of 9% for the validation set.

4.6.3 Varying the Number of Hidden Nodes. The next logical step was to vary the number of hidden nodes in the ANN design. As the number of hidden nodes was varied, a comparison was made of the classification error rate of the solution occurring at the lowest level of the total absolute error to the classification error rate at the solution occurring at the lowest levels of the previously evaluated stopping criteria (such as the classification error rate of the testing set). A closure in the form of a recommended method by which to determine the proper time to terminate ANN

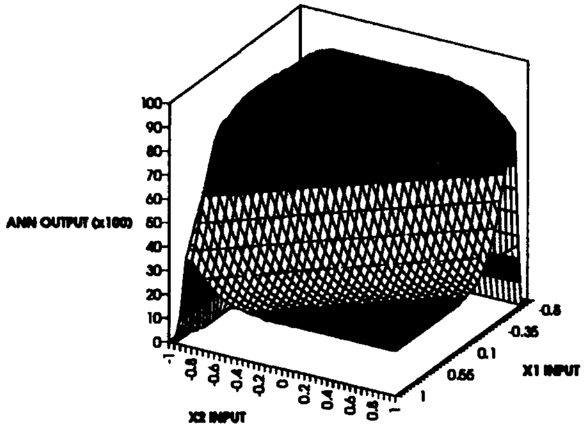


Figure 26. Mesh Solution Space: Three Hidden Nodes 50,000 Epochs

training would result if one stopping criterion proved itself superior by consistently recommending the lowest classification error rate of the validation set. A relatively small variance of the classification error rate for any stopping criterion selected at this step would further support its selection as an optimal stopping criterion.

The number of hidden nodes was varied at levels of all even numbers between four and twenty for 15,000 epochs. Potential stopping criteria measures compared to the minimum total absolute error included the following: classification error rate on the training set, classification error rate on the testing set, a evenly weighted average of the classification error rate of the training and testing sets, and a moving average of the classification error rate on the testing set over the most recent five epochs. Full results of the experiments for the ten replications at each selected number of hidden nodes can be found in Appendix D.

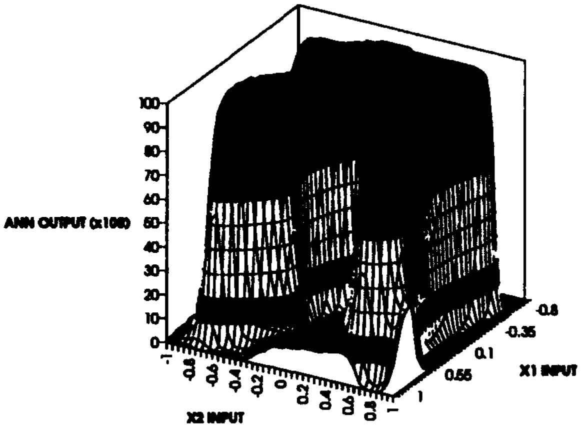


Figure 27. Mesh Solution Space: Eight Hidden Nodes at 9,000 Epochs

A detailed review of Appendix D reveals a complication in the results. A simple average was not a fair comparison. For example, if the classification error rate on the training set for a given replication were to remain much higher than that for the other replications, common sense indicated that that particular set of ending ANN weights should not be selected. Therefore, fairness dictated that these weights not be counted when computing averages. Additionally, a comparison across the different metrics revealed that some sets of starting weights led to a relatively low classification error rate on the validation set regardless of the stopping criterion used and some starting weights led to a relatively high classification error rate across all of the potential stopping criteria. This seemed to indicate an advantage of looking at more than one criterion when attempting to determine when to terminate ANN training.

Therefore, the 'obvious' outliers (indicating weights not moving toward the global optima) had to be eliminated. This section of the experiment set up a heuristic for distinguishing between those points 'good' enough to be considered and those points so 'poor' that they should be eliminated. The following heuristic was used.

1. Select the 'best' level within each potential stopping criterion. The best level was defined as the lowest total absolute error or the lowest classification error rate for the training set, testing set, and weighted average and moving average criteria. Select at least two potential points because of the weakness of selecting only one point as indicated in Figure 21 where the lowest total absolute error did not indicate the lowest classification error rate on the validation set.
2. Select approximately a 2% to 4% rate window from the 'best' classification error rate (2 to 4 in absolute value with respect to total absolute error). Actual selection depended upon the grouping of the output. The goal was to obtain a set of points which contained the best classification error rate which would be obtained if a given stopping criterion was used.
3. Using the selected replications, compute the mean and standard deviation of the classification error rate on the validation set.

4.6.4 Results: Varying the Number of Hidden Nodes. Using the above rules, the resulting classification error rates at each of the various hidden nodes tested are summarized in Table 5.

Table 5: Mesh Problem: Selected Solutions

Stopping Criterion	Average Classification Error	Standard Deviation of Classification Error
Four Hidden Nodes		
Training Set	0.2294	0.0142
Testing Set	0.2217	0.0236
Weighted Average	0.2204	0.0025
Moving Average	0.2255	0.0187
Total Absolute Error	0.2230	0.0263
Six Hidden Nodes		
Training Set	0.1588	0.0270
Testing Set	0.1570	0.0367
Weighted Average	0.1508	0.0351
Moving Average	0.1458	0.0339
Total Absolute Error	0.1479	0.0320
Eight Hidden Nodes		
Training Set	0.1267	0.0390
Testing Set	0.0967	0.0118
Weighted Average	0.0992	0.0082
Moving Average	0.1000	0.0094
Total Absolute Error	0.1000	0.0071

Table 5: Mesh Problem: Selected Solutions (continued)

Stopping Criterion	Average Classification Error	Standard Deviation of Classification Error
Ten Hidden Nodes		
Training Set	0.0894	0.0386
Testing Set	0.0861	0.0067
Weighted Average	0.0917	0.0058
Moving Average	0.0900	0.0033
Total Absolute Error	0.0883	0.0000
Twelve Hidden Nodes		
Training Set	0.1253	0.0322
Testing Set	0.0933	0.0101
Weighted Average	0.0944	0.0035
Moving Average	0.1025	0.0059
Total Absolute Error	0.0817	0.0044
Fourteen Hidden Nodes		
Training Set	0.1504	0.0195
Testing Set	0.1333	0.0284
Weighted Average	0.1278	0.0231
Moving Average	0.1194	0.0210
Total Absolute Error	0.1183	0.0176

Table 5: Mesh Problem: Selected Solutions (continued)

Stopping Criterion	Average Classification Error	Standard Deviation of Classification Error
Sixteen Hidden Nodes		
Training Set	0.1381	0.0288
Testing Set	0.1133	0.0145
Weighted Average	0.1120	0.0239
Moving Average	0.0971	0.0103
Total Absolute Error	0.1100	0.0218
Eighteen Hidden Nodes		
Training Set	0.1254	0.0146
Testing Set	0.0950	0.0180
Weighted Average	0.0977	0.0163
Moving Average	0.0937	0.0122
Total Absolute Error	0.0900	0.0115
Twenty Hidden Nodes		
Training Set	0.1117	0.0234
Testing Set	0.0850	0.0153
Weighted Average	0.0906	0.0154
Moving Average	0.0872	0.0186
Total Absolute Error	0.0833	0.0203

The actual impact of each of the examined potential stopping criteria can best be seen in a graph of the two standard deviation dispersion area of the classification error rate which would have been obtained if the lowest level of each stopping criteria were used. What follows is a graph for each ANN design as the number of hidden nodes was varied at even numbers from four to twenty (inclusive). The rationale for this decision was that the user must first make a design decision, then train the network to the lowest level for the selected stopping criterion. Next, a different design could be chosen before conducting training again. Only after the designs had been trained to an optimal level would a decision be made as to the appropriate ANN design. Therefore, the only appropriate comparison between the examined potential stopping criteria occurred when the number of hidden nodes was held constant. See Figure 28 through Figure 36 for a graphical comparison of the stopping criteria for each respective level of hidden nodes.

Given this constraint of looking at each figure separately, to goal was to determine whether any of the remaining stopping criteria consistently gave the lowest classification error rate across all of the ANN designs. The expected range over which this classification error rate would vary was also considered important. A lower variance of the classification error rate of any selected stopping criterion would further support its selection as an optimal stopping criterion. Specifically, given the earlier promise of the total absolute error stopping criterion as exhibited during the examination of the XOR problem, the question was whether or not the total absolute error would again lead to the lowest classification error rate with the minimum variance.

Comparing the diagrams, total absolute error compared favorably with the previously used criteria. In other words, it does not do any worse than the previously used criteria. However, total absolute error did not dominate those criteria such that choosing among the weights corresponding to the lowest total absolute error guaranteed the lowest classification error rate. By using the total absolute error to select a point to terminate ANN training, the resulting error classification rate was essentially the same as that obtained by using the classification error rate on the testing set, the weighted average classification error rate, or the moving average classification error rate (on the testing set) stopping criteria to terminate ANN training. However, note that the weighted average classification error rate consisted partially of the classification error rate on the testing set, which tended to drift higher as the number of hidden nodes increased.

A final point to remember regarding Figure 28 through Figure 36 is that the actual classification error rate of the population (the validation set for purposes of this study) would not be known for the general case. In this study, perfect knowledge was available and, using that perfect knowledge, an ANN design of either eight, ten, or twelve hidden nodes appeared optimal. This was based upon two facts: the classification error rate and the variance of that classification error rate were lower. Using the total absolute error as the means of selecting the point at which to terminate ANN training in these three situations would have resulted in the lowest classification error rate and variance of the classification error rate in two of the three cases.

For a different comparison of the output of the mesh problem as the number of hidden nodes was varied, see Figure 37 through Figure 41. These figures show the change in the classification error rate for each potential stopping criterion as the number of hidden nodes in varied at even numbers between four and twenty (inclusive). Recall that the mesh problem could ideally be separated by eight bounding hyperplanes. From these results, the following were noted:

- Observing across the figures, all functions demonstrated to an 'S' shaped curve of the classification error rate on the validation set. The classification error rate decreased as the number of number of hidden nodes was increased from four to eight. Next, the classification error rate remained at its lowest level for when eight, ten, or twelve hidden nodes were used. From fourteen to twenty hidden nodes, the classification error rate increased and then decreased. Using the results of these figures, an ANN consisting of eight, ten, or twelve hidden nodes would be selected.

- At the preferred design of eight, ten, or twelve hidden nodes, a lower variance of the resulting classification error rate on the validation set was observed.
- The bottom line was that the ideal ANN had a logical interpretation: using approximately eight to twelve hidden nodes resulted in the lowest mean error classification error rate on the validation set. Additionally, this ANN termination point also exhibited the lowest variance of the resulting classification error rate on the validation set.

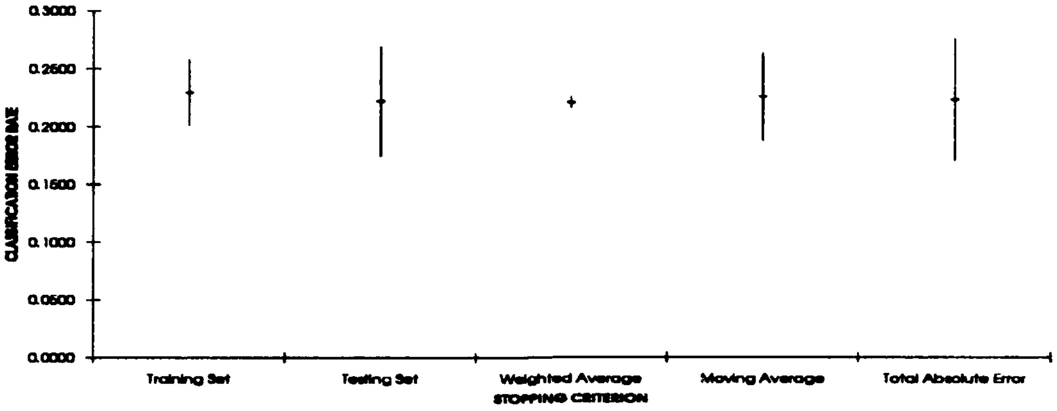


Figure 28. Mesh Classification Error Rate: Four Hidden Nodes

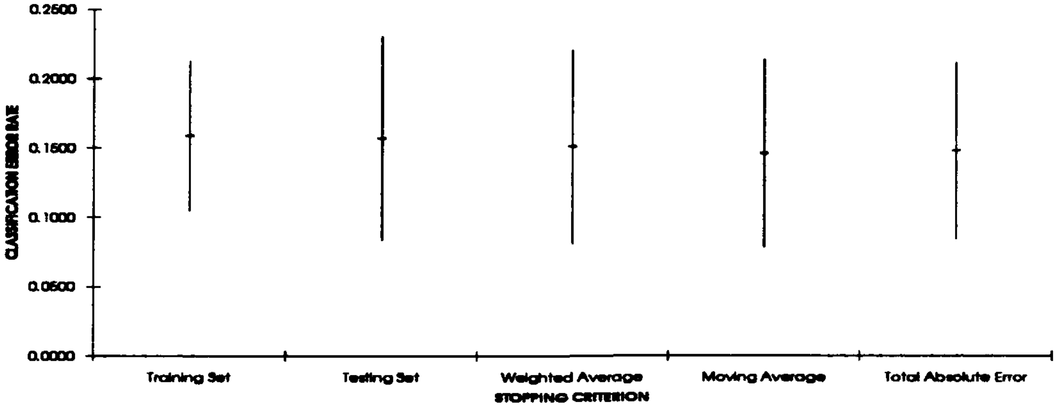


Figure 29. Mesh Classification Error Rate: Six Hidden Nodes

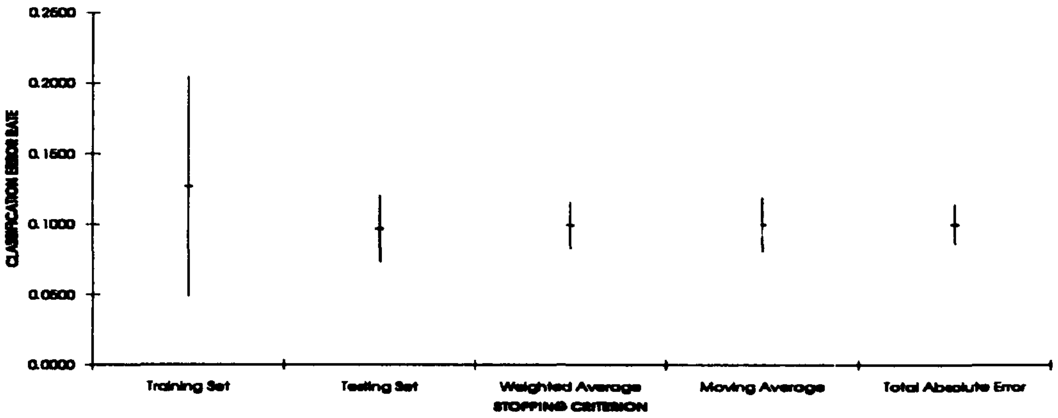


Figure 30. Mesh Classification Error Rate: Eight Hidden Nodes

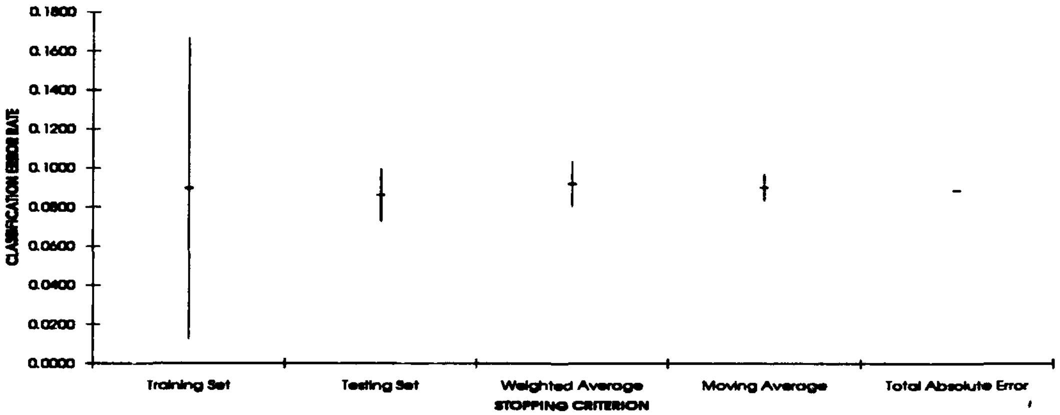


Figure 31. Mesh Classification Error Rate: Ten Hidden Nodes

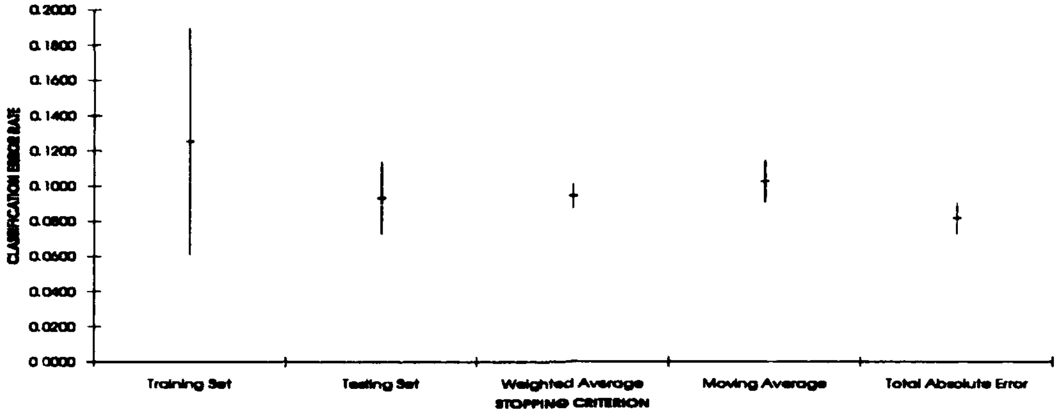


Figure 32. Mesh Classification Error Rate: Twelve Hidden Nodes

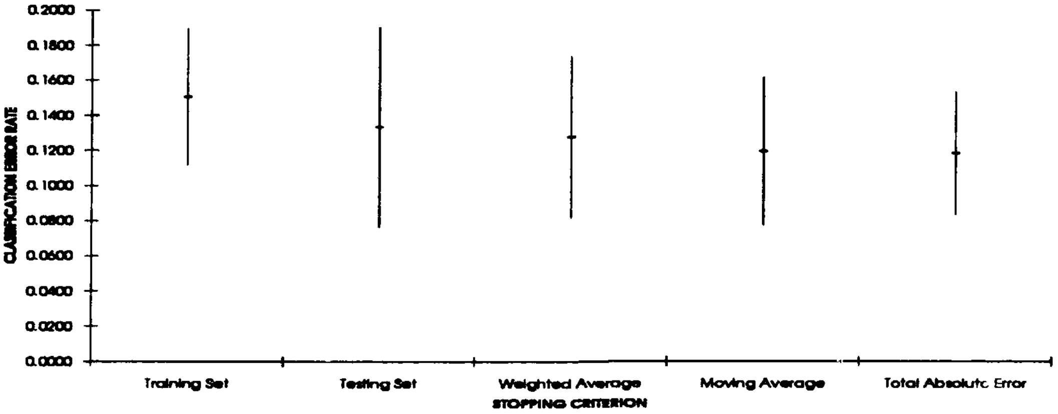


Figure 33. Mesh Classification Error Rate: Fourteen Hidden Nodes

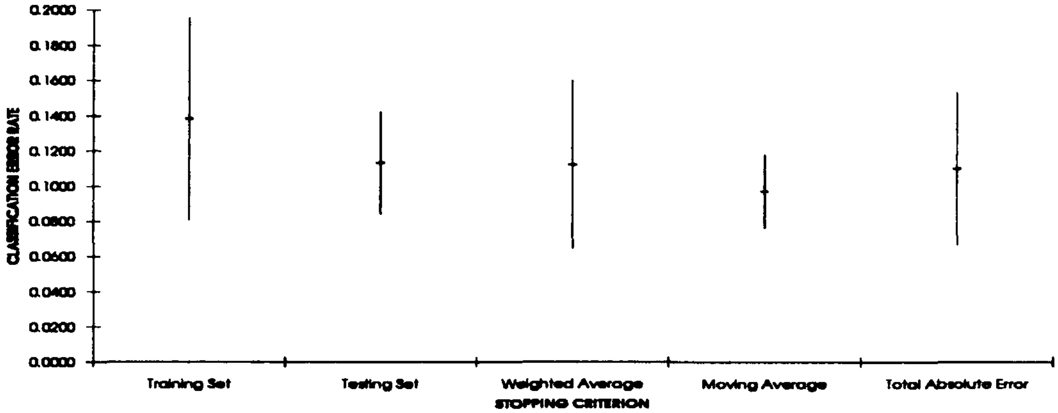


Figure 34. Mesh Classification Error Rate: Sixteen Hidden Nodes

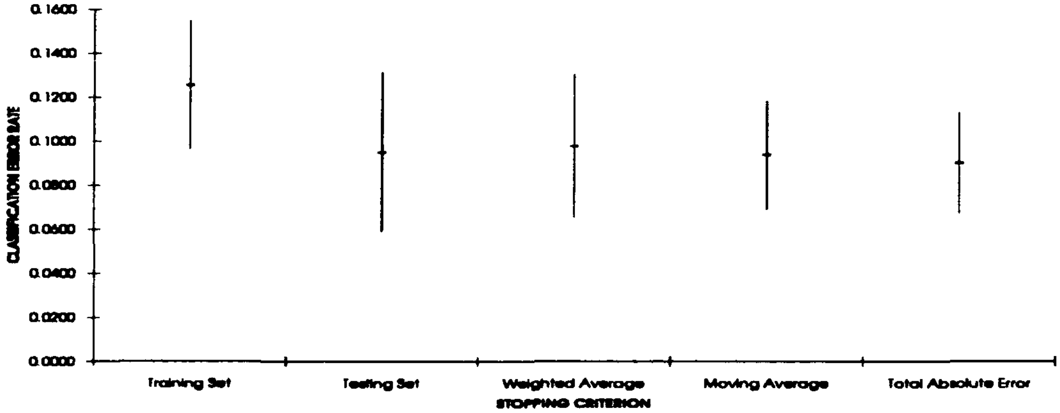


Figure 35. Mesh Classification Error Rate: Eighteen Hidden Nodes

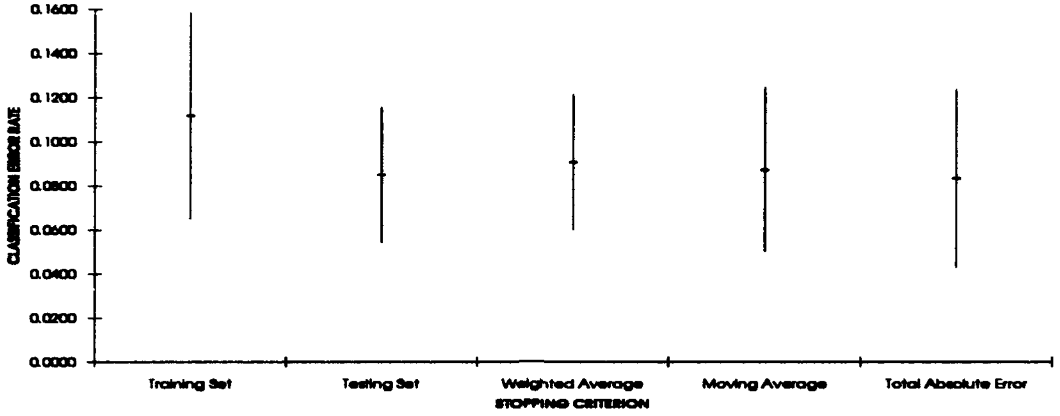


Figure 36. Mesh Classification Error Rate: Twenty Hidden Nodes

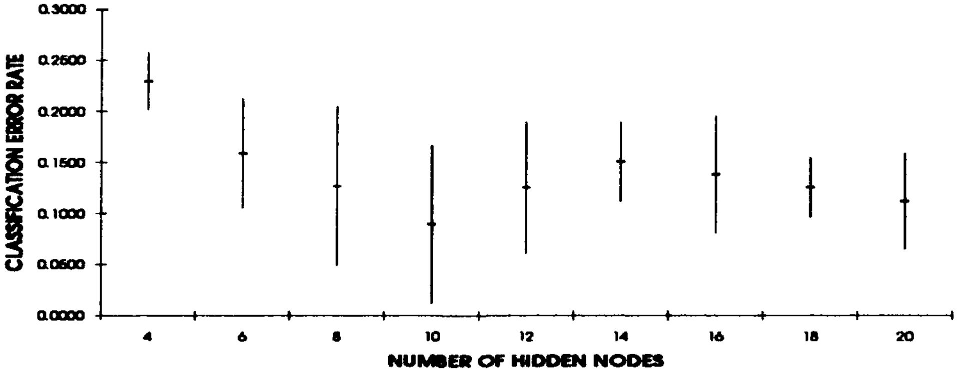


Figure 37. Mesh Problem: Training Set as Stopping Criterion

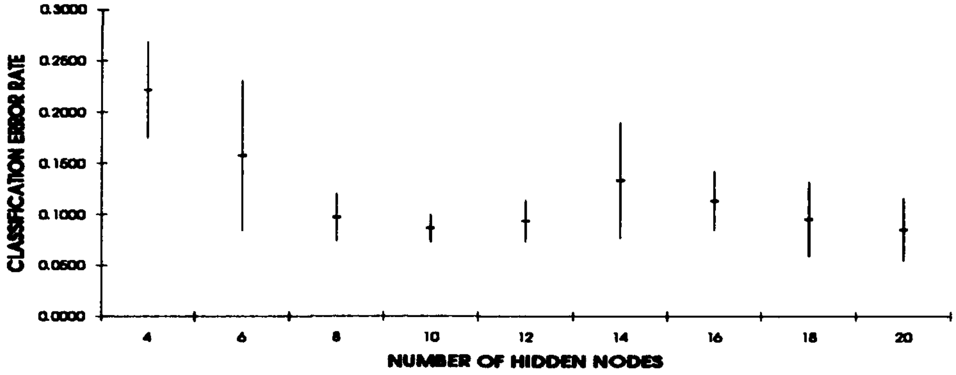


Figure 38. Mesh Problem: Testing Set as Stopping Criterion

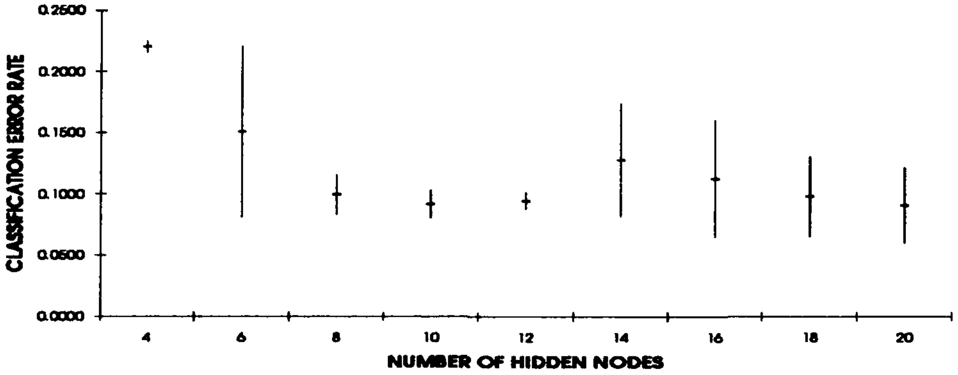


Figure 39. Mesh Problem: Weighted Average as Stopping Criterion

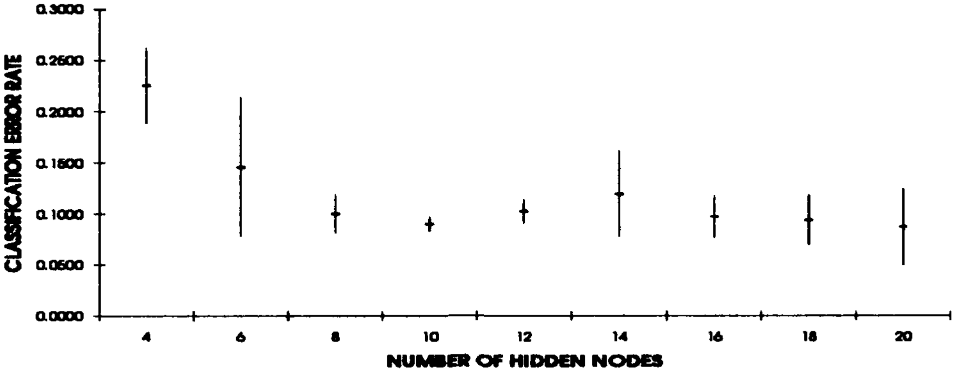


Figure 40. Mesh Problem: Moving Average as Stopping Criterion

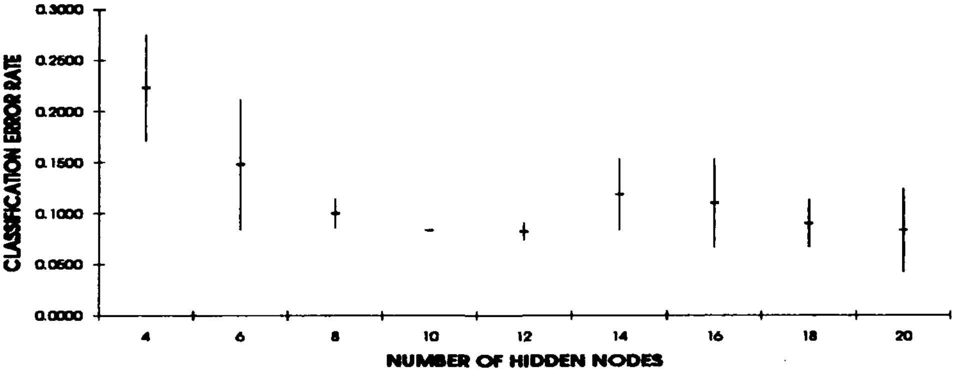


Figure 41. Mesh Problem: Total Absolute Error as Stopping Criterion

V. Results and Conclusions

5.1 A Successful Procedure

This study evaluated eleven characteristics of an artificial neural network (ANN) throughout training to determine the proper epoch at which to terminate ANN training. Based upon these characteristics, the following procedure resulted in selecting a final network which yielded a low classification error rate and also yielded a low classification error rate variance.

1. Perform at least ten replications of artificial neural network training using different random initial weights. Use the output of this sequence of experiments to determine when improvement or learning no longer occurs.
2. Perform artificial neural network training again using repeated cycles of an ANN with different random initial weights. Perform training only until the epoch at which learning no longer occurs as previously determined. Track the following functions as they occur in the testing set during this step: total absolute error, testing set classification error rate, and moving average of the testing set classification error rate.
3. Heuristically select those weights which perform well under the above three functions. Select a general grouping of the best of these, choosing at least two sets of weights for each stopping criterion. An acceptable ANN replication does well under all three categories. Generally, the area of the epoch and the initial weights will be acceptable across all three stopping criteria if the weights indicate the lowest level of classification error rate.
4. A random set of weights can then be selected from those weights designated as acceptable. Generally, this set of weights proved to yield a low classification error rate and a low variance of classification error rate. However, using this

procedure did not guarantee the lowest classification error rate on the validation set.

5.2 Potential Future Research

As with almost any area of research, progress leads toward more questions. The following extensions of this study are possible.

- Changing variables in the network to determine the impact. This study fixed a number of ANN attributes. If these were changed, the impact upon the conclusion is unknown. For example, the following could be changed: η (eta) or learning rate; the number of hidden layers; or values for the momentum term. Additionally, the interaction of these variables would seem to be of interest.
- Adding a momentum value. Since this study did not emphasize the speed at which a final ANN was determined, momentum was not in the ANN equation. The impact of adding it to the ANN equation is therefore unknown.
- Changing the maximum number of epochs. Early in the experiment process of this study, an assumption was made that ANN learning did not occur after epoch 600 for the particular network design of the XOR problem. However, the total absolute error on the testing set appeared to decrease slightly even after epoch 600. The impact of allowing the learning process to continue until the true minimum total absolute error is unknown. Particularly interesting would be the classification error rate at this final minimum.
- Using a variable momentum value. A variable momentum value is currently being researched and its impact upon the total absolute error is unknown.
- A more detailed study of overlearning. Two ANN techniques, the weighted average classification error rate and random exemplar assignment to either the training or the testing set, were designed to counteract the impact of the ten-

dency of ANN to overlearn. Since overlearning was not observed in this study, the usefulness of these tools is unknown. An examination of the techniques when overlearning occurs would be interesting.

5.2.1 Conclusion. This paper presented a technique which added insight for selecting the final ANN weights. It showed how looking at more than one function to analyze an ANN provided increased confidence that a given set of starting weights approached an optimal solution. Additionally, since a uniformly low classification error rate did not occur across all replications during ANN training, the study revealed the importance of using different initial weights and repeated replications of ANN training. In the final experiment, the number of hidden nodes was varied. This resulted in an example indicating that the ideal ANN structure (from eight to twelve hidden nodes for the mesh problem) exhibited a low classification error rate on the validation set and exhibited a low variance of that classification error rate. These respective low levels occurred for all three of the final stopping criteria judged best: classification error rate of the testing set, a moving average of the classification error rate on the testing set, and the total absolute error on the testing set. Finally, future extensions of the research for an improvement of this procedure were presented which show much room for future effort in the area of artificial neural networks.

Appendix A. FORTRAN Program

```
**********************************************************************
*
*                                                    AS OF:  14 Feb 94
*
*    The following program is a modified version of a program written
*    by Lt Col Kenneth W. Bauer in April 1993.  It uses the back
*    propagation algorithm as documented by Lippmann and Rogers.
*
**********************************************************************
*
*    Variables in this program have the following meanings:
*
*     a:         temporary storage for sigmoid function value
*     abserrst:  level to stop training for total absolute error criterion
*     arrbig:    array of combined training and testing set
*     arrtng:    array of training set exemplars
*     arrtst:    array of testing set exemplars
*     arrval:    array of validation set exemplars
*     awcstop:   level to stop training for the absolute weight change
*                criterion
*     ep:        epsilon, or the error. Lippmann's (d(j) - y(j)) term
*     errlrg:    level to stop training for the largest absolute error
*                criterion
*     essstop1:  level to stop training for the error sum of squares
*                criterion
*     errstop2:  level to stop training for the weighted average criterion
*     errsttng:  level to stop training for training set criterion
*     errsttst:  level to stop training for testing set criterion
*     errstval:  level to stop training for validation set criterion
*     eta:       gain term or step size
*     iaberr:    switch to have the program test for absolute error
*     icont1:    number of continuous epochs in which a characteristic
*                must occur to trip a stopping criterion threshold
*     icount:    item number of the examplar currently under evaluation
*     iepoch:    number of epochs conducted
*     intervl:   length of interval for moving average computations
*     istop:     maximum number of iterations or epochs
*     itest:     switch to have the program test the test set
*     itrain:    switch to have the program test the training set
*     ival:      switch to have the program test the validation set
*     m:         number of inputs or features
*     mastop:    level to stop training for moving average criterion
*     mwcstop:   level to stop training for the mean absolute relative
*                weight change criterion
*     n:         number of hidden units
*     ncstop:    number of ncycle before termination
*     ncycle:    number of times the complete program has run
*                (with new weights for each iteration)
*     no:        number of output units
*     ntngpt:    number of data points in the training set (examplars)
*     ntstpt:    number of data points in the testing set
*     nvalpt:    number of data points in the validation set
*     o:         number of output units = no
*     rwcstop:   level to stop training for the absolute relative weight
*                change criterion
*     sum:       Lippmann's delta(j) term for updating internal hidden mode
*                weights
*     swcstop:   level to stop training for the squared weight change
*                criterion
*     toterr:    total error for one examplar and one epoch
*     w1:        layer 1 weights, i to j.  i = m+1 for the bias term
*     w2:        layer 2 weights, i to j.  i = n+1 for the bias term
*     xo:        input variables, to include bias term (1) and desired output
*     x1:        hidden layer outputs
*     xx:        array value used for mapping fort.10
*     zz:        computed value of hidden nodes.
*
***********************************************************************

      program ProgFin1

*  Set neural net parameters and the size of the exemplar files

      parameter(m=2,n=3,no=1,istop=500)
      parameter(intervl=5)
      parameter(icont1=2)
      parameter(ncstop=2)
      parameter(ntngpt=100)
      parameter(ntstpt=100)
      parameter(nvalpt=600)

***** if ioption = 0, then I am keeping the training and test sets separate
***** if ioption = 1, then I mix the training and test sets between epochs

      parameter(ioption=0)

***** Parameters required for Shell-Mezgar sort routine

      PARAMETER(ALN2I=1.0/0.69314718,TINY=1.0E-5)

* Initialize variables

      real x0(m+3),x1(n+1),zz(no),ep(no)
      real w1(m+1,n),w2(n+1,no)
      real w1temp(m+1,n), w2temp(n+1,no)
      real arrbig(m+3, 2*(ntngpt+ntstpt))
      real arrtng(m+3,ntngpt), arrtst(m+2,ntstpt), arrval(m+2, nvalpt)
      real mwcstop, mastop
      integer xx(11,11)
      integer o
      real movavg(intervl), avg(intervl), rate(intervl)

***** Variables required for Shell-Mezgar sort routine

      real rndm(ntngpt)
      real rndm2(ntngpt+ntstpt)

*   istXXX stands for i stop for reason XXX.

      integer isttst(ncstop), isttrn(ncstop), istval(ncstop)
      integer istcyc(ncstop), istwa(ncstop)
      integer isttae(ncstop), isttaeb(ncstop), itemp1(ncstop)
      integer istess(ncstop), istessb(ncstop), itemp2(ncstop)
      integer istlae(ncstop), istlaeb(ncstop), itemp3(ncstop)
      integer istawc(ncstop), istswc(ncstop), istrwc(ncstop)
      integer istmwc(ncstop), istma(ncstop), isttrnb(ncstop)
      integer itempa(ncstop), itempb(ncstop), itempc(ncstop)
      integer itempd(ncstop), itempe(ncstop), itempf(ncstop)
      integer itempg(ncstop)
      integer isttstb(ncstop), istwab(ncstop), istrwcb(ncstop)
      integer isttemp(ncstop)

*   igrafXX is for graph switch number XX

      integer igraf01(ncstop), igraf02(ncstop), igraf03(ncstop)
      integer igraf04(ncstop), igraf05(ncstop), igraf06(ncstop)
      integer igraf07(ncstop), igraf08(ncstop), igraf09(ncstop)
      integer igraf10(ncstop), igraf11(ncstop), igraf12(ncstop)
      integer igraf13(ncstop), igraf14(ncstop), igraf15(ncstop)
      integer igraf16(ncstop), igraf17(ncstop), igraf18(ncstop)
      integer igraf19(ncstop), igraf20(ncstop), igraf21(ncstop)
      integer igraf22(ncstop), igraf23(ncstop), igraf24(ncstop)
      integer igraf25(ncstop)
      integer igraf31(ncstop), igraf32(ncstop), igraf33(ncstop)
      integer igraf34(ncstop), igraf35(ncstop), igraf36(ncstop)
      integer igraf37(ncstop), igraf38(ncstop), igraf39(ncstop)

*   Set up the weighted average percentages

      data pertrn/.50/
      data pertst/.50/
      data perval/0.0/

*   Set gain term or step size in eta and the percent stopping criteria

      data eta/.35/
```

```
*  Set the parameters to force a stop of training at the following
*  threshold levels

      data errsttng/1.0/
      data errsttst/1.0/
      data errstval/1.0/
      data errstop2/1.0/
      data abserrst/4.0/
      data essstop1/1.0/
      data errlrg/.5/
      data awcstop/.04/
      data swcstop/.0003/
      data rwcstop/.001/
      data mwcstop/.004/
      data mastop/.01/

***** Constants required for Shell-Mezgar sort routine

      LOGNB2=INT(ALOG(FLOAT(ntngpt))*ALN2I+TINY)
      LOGNB3=INT(ALOG(FLOAT(ntngpt+ntstpt))*ALN2I+TINY)

      o=no

* iprint always = 1

      iprint=1

*  Switches for determining which set of data to test against

         itest = 1
         itrain = 1
         ival = 1
         iaberr = 1

*  Initialize switches for determining reason program terminated

      do 509 iii = 1, ncstop
         istcyc(iii) = 0
         isttst(iii) = 0
         isttrn(iii) = 0
         istval(iii) = 0
         istwa(iii) = 1
         istwab(iii) = 1
         isttae(iii) = 0
         isttaeb(iii) = 0
         itemp1(iii) = 0
         istess(iii) = 1
         istessb(iii) = 1
         itemp2(iii) = 0
         istlae(iii) = 0
```

```
istlaeb(iii) = 0
itemp3(iii) = 0
istawc(iii) = 1
istswc(iii) = 1
istrwc(iii) = 0
istrwcb(iii) = 0
istmwc(iii) = 1
istma(iii) = 0
isttrnb(iii) = 0
isttstb(iii) = 0
itempa(iii) = 0
itempb(iii) = 0
itempc(iii) = 0
itempd(iii) = 0
itempe(iii) = 0
itempf(iii) = 0
itempg(iii) = 0
igraf01(iii) = 0
igraf02(iii) = 0
igraf03(iii) = 0
igraf04(iii) = 0
igraf05(iii) = 0
igraf06(iii) = 0
igraf07(iii) = 0
igraf08(iii) = 0
igraf09(iii) = 0
igraf10(iii) = 0
igraf11(iii) = 0
igraf12(iii) = 0
igraf13(iii) = 0
igraf14(iii) = 0
igraf15(iii) = 0
igraf16(iii) = 0
igraf17(iii) = 0
igraf18(iii) = 0
igraf19(iii) = 0
igraf20(iii) = 0
igraf21(iii) = 0
igraf22(iii) = 0
igraf23(iii) = 0
igraf24(iii) = 0
igraf25(iii) = 0
igraf31(iii) = 0
igraf32(iii) = 0
igraf33(iii) = 0
igraf34(iii) = 0
igraf35(iii) = 0
igraf36(iii) = 0
igraf37(iii) = 0
igraf38(iii) = 0
igraf39(iii) = 0
```

```
         isttemp(iii) = 0
509   continue

*  read the exemplars into arrays
*       THIS SECTION HAS TO CHANGE IF THERE ARE OTHER THAN FOUR COLUMNS.
*  fort.22 contains the original training vectors
*  fort.7 contains the original testing vectors
*  fort.9 contains the original validation vectors

      do 21 iii = 1, ntngpt
         read(22,*) arrtng(1, iii), arrtng(2, iii),
     *              arrtng(3, iii), arrtng(4, iii)
21    continue

      do 22 iii = 1, ntstpt
         read(7,*) arrtst(1, iii), arrtst(2, iii),
     *              arrtst(3, iii), arrtst(4, iii)
22    continue

      do 23 iii = 1, nvalpt
         read(9,*) arrval(1, iii), arrval(2, iii),
     *              arrval(3, iii), arrval(4, iii)
23    continue

*  read in the training and testing exemplars into one big array.

      rewind 22
      rewind 7

      do 24 iii = 1, ntngpt
         read(22,*) arrbig(1, iii), arrbig(2, iii),
     *               arrbig(3, iii), arrbig(4, iii)
         read(7,*) arrbig(1, ntngpt+iii), arrbig(2, ntngpt+iii),
     *              arrbig(3, ntngpt+iii), arrbig(4, ntngpt+iii)
24    continue


*  Set ibdg = 1 to get a file of the audit trail of the program.

      idbg=0

      write (*,*)'# inputs, # hidden units, # output units'
      write (*,*)m,n,no
      write (*,*)'# exemplars, # epochs'
      write (*,*)ntngpt,istop
      write (*,*)'step size'
      write (*,*)eta
      if (idbg.eq.1) then
         write (*,*) 'Debug writes in fort.2'
      endif
      write (*,*) 'Final weights and net predictions in fort.2'
      write (*,*) 'Activation Map in fort.10'
      write (*,*) 'Class Rate on Train Set in fort.11'
      write (*,*) 'Class Rate on Test Set in fort.12'
      write (*,*) 'Class Rate on Val Set in fort.13'
      write (*,*) 'Tot Absolute Err on Test Set in fort.14'
      write (*,*) 'Tot Err Sum of Squar on Test Set in fort.15'
      write (*,*) 'Largest Absolute Error on Test Set in fort.16'
      write (*,*) 'Weighted Average Error Rate in fort.17'
      write (*,*) 'Moving average calculations in fort.18'
      write (*,*) 'Absolute Weight Change in fort.41'
      write (*,*) 'Squared Weight Change in fort.42'
      write (*,*) 'Relative Weight Change in fort.43'
      write (*,*) 'Mean Weight Change in fort.44'
      write (*,*) 'Summary writes in fort.50'
      write (*,*) 'Stopping points write in fort.60 and on'

      write (2,*)'(x0(k),k=1,m),x0(m+2),zz(1)'
      write (50,*) 'SUMMARY FILE'
      write (50,*) 'EPOCH, ERR RATE ON VAL SET, REASON'
      write (50,*) ' '
      write (50,*) 'STARTING CYCLE NUMBER ', ncycle + 1
      write (50,*) ' '

*  Initialize the randum number generator seed to any negative number

      idum = (-1)
      idum2 = (-2)
      idum0 = (-3)

*  Initialize the criteria for number of cycles to run

      ncycle = 1

*  Continue if there is not enough cycles to stop.

 501  write (2,*) ' '
      write (2,*) ' '
      write (2,*) 'WEIGHTS FOR CYCLE NUMBER  ', ncycle

      continue

*  randomly initialize the weights to include weights for the nodes and
*  weights for the bias term.  Bias term is the reason for m+1 and
*  n+1 instead of m and n respectively.
*  Use numwts to count the number of weights.

      numwts = 0

      do 100 i=1,m+1
       do 100 j=1,n
         w1(i,j)=rano(idum)
         w1temp(i,j) = w1(i,j)
         numwts = numwts + 1
100   continue
      do 101 i=1,n+1
      do 101 j=1,o
         w2(i,j)=rano(idum)
         w2temp(i,j) = w2(i,j)
         numwts = numwts + 1
101   continue

      iepoch=0
1     icount=0

*  BEGIN LOOP TO RANDOMLY REORDER THE TRAINING EXEMPLARS

***** Randomly sort training vectors for input to network.  RNDM is used to
***** hold random numbers.  These random numbers are then paired with the
***** training vectors.  A Shell-Mezgar sort is then performed using RNDM as
***** the sort key.  This yields a random ordering of the training vectors.
***** THIS ROUTINE WAS CODED BY GREG REINHART.

      if (ioption.eq.1) then
         go to 9001
      else
         continue
      endif

        DO 1912 I=1,ntngpt
           RNDM(I)=RANO(RANDOM)
1912      CONTINUE
        Mgreg=ntngpt
        DO 1915 NN=1,LOGNB2
           Mgreg=Mgreg/2
           K=ntngpt-Mgreg
           DO 1913 J=1,K
              I=J
1917            L=I+Mgreg
              IF(RNDM(L).LT.RNDM(I)) THEN
                temp9=RNDM(I)
                RNDM(I)=RNDM(L)
                RNDM(L)=temp9
                DO 1920 MM=1,m+2
                   temp9=arrtng(I,MM)
                   arrtng(I,MM)=arrtng(L,MM)
                   arrtng(L,MM)=temp9
1920              CONTINUE
                I=I-Mgreg
                IF(I.GE.1) GOTO 1917
              ENDIF
1913         CONTINUE
1915      CONTINUE
```

```
9001  continue

      if (ioption.eq.0) then
         go to 9002
      else
         continue
      endif

        DO 9912 I=1,ntngpt + ntstpt
           RNDM2(I)=RANO(RANDOM)
9912      CONTINUE
        Mgreg=ntngpt+ntstpt
        DO 9915 NN=1,LOGNB3
           Mgreg=Mgreg/2
           K=ntngpt+ntstpt-Mgreg
           DO 9913 J=1,K
              I=J
9917            L=I+Mgreg
              IF(RNDM(L).LT.RNDM(I)) THEN
                temp9=RNDM(I)
                RNDM(I)=RNDM(L)
                RNDM(L)=temp9
                DO 9920 MM=1,m+2
                   temp9=arrbig(I,MM)
                   arrbig(I,MM)=arrbig(L,MM)
                   arrbig(L,MM)=temp9
9920              CONTINUE
                I=I-Mgreg
                IF(I.GE.1) GOTO 9917
              ENDIF
9913         CONTINUE
9915      CONTINUE

9002  continue

*  END LOOP TO REORDER TRAINING EXEMPLARS

      iepoch=iepoch+1

*  iswitch tells if a stopping criterion has been met this epoch

      iswitch = 0

2     icount=icount+1

* debug loop
      if(idbg.eq.1)then
        write(2,*)'input data --- icount --- iepoch'
        write(2,*)(x0(i),i=1,m+2),icount,iepoch
      endif
      if (ioption.eq.1) then
         go to 9003
      else
         continue
      endif

*  FEEDFORWARD:  for each input node, compute an output on the first
*  layer x1(ii) and an input to the hidden layer.
*  use m+1 to include the bias term

      do 10 ii=1,n
       a=0.
      do 20 ll=1,m+1
20     a=a+w1(ll,ii)*arrtng(ll,icount)
10    x1(ii)=f(a)

* debug loop
      if(idbg.eq.1)then
        write(2,*)'Hidden Layer Outputs'
        write(2,*)x1
      endif

* set up bias term for hidden layer

      x1(n+1)=1.0

* continuation of feedforward loop.  compute an output term zz(jj)

      do 30 jj=1,o
       a=0.
      do 40 ii=1,n+1
40     a=a+w2(ii,jj)*x1(ii)
30    zz(jj)=f(a)

* debug loop
      if(idbg.eq.1)then
        write(2,*)'Output Layer Outputs'
        write(2,*)zz
      endif

*  COMPUTE ERROR.  x0(m+2) is the input desired answer.

      do 50 jj=1,o
50     ep(jj)=arrtng(m+2,icount)-zz(jj)

* debug loop
      if(idbg.eq.1)then
        write(2,*)'LAYER 1 WEIGHTS ---- BEFORE UPDATES'
        do 300 i=1,m+1
300      write(2,131) (w1(i,j),j=1,n)
```

```
        write(2,*)'LAYER 2 WEIGHTS ---- BEFORE UPDATES'
        do 301 i=1,n+1
301      write(2,131)(w2(i,j),j=1,o)
      endif

131       format(1x,8f8.3)

*  OUTPUT LAYER WEIGHT UPDATE.
*  the term ep(jj)*(1.-zz(jj))*zz(jj) is Lippmann's delta(j) term

      x1(n+1)=1.0
      do 60 ii=1,n+1
      do 60 jj=1,o
60     w2(ii,jj)=w2(ii,jj)+ep(jj)*(1.-zz(jj))*zz(jj)*x1(ii)*eta

*  HIDDEN LAYER WEIGHT UPDATE.
*  the term ep(jj)*(1.-zz(jj))*zz(jj)*w2(ii,jj) is Lippmann's term delta(k),
*       with the sum part used to make a summation.
*  x0(ll) is Lippmann's term x^prime(i).
*  the term sum*(1.-x1(ii))*x1(ii) is Lippmann's term delta(j).

      do 80 ii=1,n
      do 80 ll=1,m+1
       sum=0.
      do 90 jj=1,o
90     sum=sum+ep(jj)*(1.-zz(jj))*zz(jj)*w2(ii,jj)
80    w1(ll,ii)=w1(ll,ii)+(1.-x1(ii))*x1(ii)*arrtng(ll,icount)*sum*eta

9003  continue

      if (ioption.eq.0) then
         go to 9004
      else
         continue
      endif

*  FEEDFORWARD:  for each input node, compute an output on the first
*  layer x1(ii) and an input to the hidden layer.
*  use m+1 to include the bias term

      do 9010 ii=1,n
       a=0.
      do 9020 ll=1,m+1
9020     a=a+w1(ll,ii)*arrbig(ll,icount)
9010    x1(ii)=f(a)

* debug loop
      if(idbg.eq.1)then
        write(2,*)'Hidden Layer Outputs'
        write(2,*)x1
      endif
```

```
* set up bias term for hidden layer

      x1(n+1)=1.0

* continuation of feedforward loop.  compute an output term zz(jj)

      do 9030 jj=1,o
       a=0.
      do 9040 ii=1,n+1
9040     a=a+w2(ii,jj)*x1(ii)
9030    zz(jj)=f(a)

* debug loop
      if(idbg.eq.1)then
        write(2,*)'Output Layer Outputs'
        write(2,*)zz
      endif

*  COMPUTE ERROR.  x0(m+2) is the input desired answer.

      do 9050 jj=1,o
9050     ep(jj)=arrbig(m+2,icount)-zz(jj)

* debug loop
      if(idbg.eq.1)then
        write(2,*)'LAYER 1 WEIGHTS ---- BEFORE UPDATES'
        do 9300 i=1,m+1
9300      write(2,131) (w1(i,j),j=1,n)
        write(2,*)'LAYER 2 WEIGHTS ---- BEFORE UPDATES'
        do 9301 i=1,n+1
9301      write(2,131)(w2(i,j),j=1,o)
      endif

*  OUTPUT LAYER WEIGHT UPDATE.
*  the term ep(jj)*(1.-zz(jj))*zz(jj) is Lippmann's delta(j) term

      x1(n+1)=1.0
      do 9060 ii=1,n+1
      do 9060 jj=1,o
9060     w2(ii,jj)=w2(ii,jj)+ep(jj)*(1.-zz(jj))*zz(jj)*x1(ii)*eta

*  HIDDEN LAYER WEIGHT UPDATE.
*  the term ep(jj)*(1.-zz(jj))*zz(jj)*w2(ii,jj) is Lippmann's term delta(k),
*       with the sum part used to make a summation.
*  x0(ll) is Lippmann's term x^prime(i).
*  the term sum*(1.-x1(ii))*x1(ii) is Lippmann's term delta(j).

      do 9080 ii=1,n
      do 9080 ll=1,m+1
       sum=0.
      do 9090 jj=1,o
9090     sum=sum+ep(jj)*(1.-zz(jj))*zz(jj)*w2(ii,jj)
9080    w1(ll,ii)=w1(ll,ii)+(1.-x1(ii))*x1(ii)*arrbig(ll,icount)*sum*eta

9004  continue

* debug loop
      if(idbg.eq.1)then
        write(2,*)'LAYER 1 WEIGHTS ---- AFTER UPDATES'
        do 400 i=1,m+1
400       write(2,131) (w1(i,j),j=1,n)
        write(2,*)'LAYER 2 WEIGHTS ---- AFTER UPDATES'
        do 401 i=1,n+1
401       write(2,131)(w2(i,j),j=1,o)
      endif

*  If all of the exemplars have been checked, continue and compute error.
*  Otherwise, take the else path and return to 2
*             the else block is toward the end of the program.

      if(ntngpt.eq.icount)then

*  If all examplars have been checked, begin checking for stopping
*         criteria.

**********************************************************************
*
*
*  BEGIN STOPPING CRITERIA CALCULATIONS
*
*
**********************************************************************

****
****   FIND CLASSIFICATION ERROR RATE FOR TRAINING SET
****

      if (ioption.eq.1) then
         go to 9005
      else
         continue
      endif

      if (itrain.eq.1) then
         nclas9 = 0
         do 1502 i1 = 1, ntngpt

*  FEEDFORWARD:  for each input node, compute an output on the first
*  layer x1(ii) and an input to the hidden layer.
*  use m+1 to include the bias term
```

```
         do 1510 ii=1,n
            b=0.
            do 1520 ll=1,m+1
1520             b=b+w1(ll,ii)*arrtng(ll,i1)
1510             x1(ii)=f(b)

* debug loop
      if(idbg.eq.1)then
        write(2,*)'Hidden Layer Outputs'
        write(2,*)x1
      endif

* set up bias term for hidden layer

         x1(n+1)=1.0

* continuation of feedforward loop.  compute an output term zz(jj)

         do 1530 jj=1,o
            b=0.
            do 1540 ii=1,n+1
1540             b=b+w2(ii,jj)*x1(ii)
                zz(jj)=f(b)

*  discretize the classification

                if (zz(jj).lt.(.5)) then
                   iclass = 0
                else
                   iclass = 1
                endif

1530      continue

*  compare to desired result

         if (iclass.eq.nint(arrtng(m+2,i1))) then
             nclas9 = nclas9 + 1
         else
             continue
         endif

1502      continue

         clasrat1 = (real(nclas9))/(real(ntngpt))

         write(11,*) iepoch, 1.0 - clasrat1
      else
         continue
      endif
```

```
9005  continue

      if (ioption.eq.0) then
         go to 900[illegible]
      else
         continue
      endif

      if (itrain.eq.1) then
         nclas9 = 0
         do 9502 i1 = 1, ntngpt

*  FEEDFORWARD:  for each input node, compute an output on the first
*  layer x1(ii) and an input to the hidden layer.
*  use m+1 to include the bias term

         do 9510 ii=1,n
            b=0.
            do 9520 ll=1,m+1
9520             b=b+w1(ll,ii)*arrbig(ll,i1)
9510             x1(ii)=f(b)

* debug loop
      if(idbg.eq.1)then
        write(2,*)'Hidden Layer Outputs'
        write(2,*)x1
      endif

* set up bias term for hidden layer

         x1(n+1)=1.0

* continuation of feedforward loop.  compute an output term zz(jj)

         do 9530 jj=1,o
            b=0.
            do 9540 ii=1,n+1
9540             b=b+w2(ii,jj)*x1(ii)
                zz(jj)=f(b)

*  discretize the classification

                if (zz(jj).lt.(.5)) then
                   iclass = 0
                else
                   iclass = 1
                endif

9530      continue

*  compare to desired result
```

```
         if (iclass.eq.nint(arrbig(m+2,i1))) then
             nclas9 = nclas9 + 1
         else
             continue
         endif

9502      continue

         clasrat1 = (real(nclas9))/(real(ntngpt))

         write(11,*) iepoch, 1.0 - clasrat1
      else
         continue
      endif

9006  continue

****
****   END:  FIND CLASSIFICATION ERROR RATE FOR TRAINING SET
****

****
****   FIND CLASSIFICATION ERROR RATE FOR TESTING SET
****

      if (ioption.eq.1) then
         go to 9007
      else
         continue
      endif

      if (itest.eq.1) then
         nclas3 = 0

         do 1503 i1 = 1, ntstpt

*  FEEDFORWARD:  for each input node, compute an output on the first
*  layer x1(ii) and an input to the hidden layer.
*  use m+1 to include the bias term

         do 1511 ii=1,n
            b=0.
            do 1521 ll=1,m+1
1521             b=b+w1(ll,ii)*arrtst(ll,i1)
1511             x1(ii)=f(b)

* debug loop
      if(idbg.eq.1)then
        write(2,*)'Hidden Layer Outputs'
        write(2,*)x1
      endif

* set up bias term for hidden layer

         x1(n+1)=1.0

* continuation of feedforward loop.  compute an output term zz(jj)

         do 1531 jj=1,o
            b=0.
            do 1541 ii=1,n+1
1541             b=b+w2(ii,jj)*x1(ii)
                zz(jj)=f(b)

*  discretize the classification

                if (zz(jj).lt.(.5)) then
                   iclass = 0
                else
                   iclass = 1
                endif

1531      continue

*  compare to desired result

          if (iclass.eq.nint(arrtst(m+2,i1))) then
             nclas3 = nclas3 + 1
          else
             continue
          endif

1503      continue

         clasrat2 = (real(nclas3))/(real(ntstpt))

         write(12,*) iepoch, 1.0 - clasrat2
      else
         continue
      endif

9007  continue

      if (ioption.eq.0) then
         go to 9008
      else
         continue
      endif

      if (itest.eq.1) then
         nclas3 = 0
         do 9503 i1 = ntngpt+1, ntngpt+ntstpt

*  FEEDFORWARD:  for each input node, compute an output on the first
*  layer x1(ii) and an input to the hidden layer.
*  use m+1 to include the bias term

         do 9511 ii=1,n
            b=0.
            do 9521 ll=1,m+1
9521             b=b+w1(ll,ii)*arrbig(ll,i1)
9511             x1(ii)=f(b)

* debug loop
      if(idbg.eq.1)then
        write(2,*)'Hidden Layer Outputs'
        write(2,*)x1
      endif

* set up bias term for hidden layer

         x1(n+1)=1.0

* continuation of feedforward loop.  compute an output term zz(jj)

         do 9531 jj=1,o
            b=0.
            do 9541 ii=1,n+1
9541             b=b+w2(ii,jj)*x1(ii)
                zz(jj)=f(b)

*  discretize the classification

                if (zz(jj).lt.(.5)) then
                   iclass = 0
                else
                   iclass = 1
                endif

9531      continue

*  compare to desired result

          if (iclass.eq.nint(arrbig(m+2,i1))) then
             nclas3 = nclas3 + 1
          else
             continue
          endif

9503      continue
```

```
         clasrat2 = (real(nclas3))/(real(ntstpt))

         write(12,*) iepoch, 1.0 - clasrat2
      else
         continue
      endif

9008  continue

****  BEGIN UPDATE OF MOVING AVERAGE OF LAST intervl OBSERVATIONS

      if (iepoch.le.intervl) then
         avg(iepoch) = 1.0 - clasrat2
         temp5 = 0.0
         do 701 ij = 1, iepoch
            temp5 = temp5 + avg(ij)
701      continue
         movavg(iepoch) = temp5/(real(iepoch))
         rate(iepoch) = 1.0 - clasrat2
      else
         continue
      endif

      if (iepoch.gt.intervl) then
         temp6 = rate(1)
*  update the rate block
         do 703 ij = 1, intervl
            rate(ij) = rate(ij + 1)
703      continue
         do 706 ij = 1, intervl
            movavg(ij) = movavg(ij + 1)
706      continue
         rate(intervl) = 1.0 - clasrat2
         movavg(intervl) = movavg(intervl - 1) + ((rate(intervl)
     *         - temp6)/(real(intervl)))
         write (18,*) iepoch, movavg(intervl)
      else
         continue
      endif


****  END UPDATE OF MOVING AVERAGE OF LAST intervl OBSERVATIONS


****
****   END:  FIND CLASSIFICATION ERROR RATE FCR TESTING SET
****


****
****   FIND CLASSIFICATION ERROR RATE FOR VALIDATION SET
```

```
****

      if (ival.eq.1) then
         nclas2 = 0

         do 1504 i1 = 1, nvalpt

*  FEEDFORWARD:  for each input node, compute an output on the first
*  layer x1(ii) and an input to the hidden layer.
*  use m+1 to include the bias term

         do 1512 ii=1,n
            b=0.
            do 1522 ll=1,m+1
1522             b=b+w1(ll,ii)*arrval(ll,i1)
1512             x1(ii)=f(b)

* debug loop
      if(idbg.eq.1)then
        write(2,*)'Hidden Layer Outputs'
        write(2,*)x1
      endif

* set up bias term for hidden layer

         x1(n+1)=1.0

* continuation of feedforward loop.  compute an output term zz(jj)

         do 1532 jj=1,o
            b=0.
            do 1542 ii=1,n+1
1542             b=b+w2(ii,jj)*x1(ii)
               zz(jj)=f(b)

*  discretize the classification

               if (zz(jj).lt.(.5)) then
                  iclass = 0
               else
                  iclass = 1
               endif

1532      continue

*  compare to desired result

         if (iclass.eq.nint(arrval(m+2,i1))) then
             nclas2 = nclas2 + 1
         else
             continue
         endif

1504      continue

         clasrat5 = (real(nclas2))/(real(nvalpt))

         write(13,*) iepoch, 1.0 - clasrat5
      else
         continue
      endif

****
****   END:  FIND CLASSIFICATION ERROR RATE FOR VALIDATION SET
****

****
****   BEGIN COMPUTING TOTAL ABSOLUTE ERROR ON TESTING SET
****   THIS LOOP ALSO COMPUTES ERROR SUM OF SQUARES
****   THIS LOOP ALSO COMPUTES LARGEST ABSOLUTE ERROR
****

      if (iaberr.eq.1) then
         temp1 = 0.0
         temp2 = 0.0
         temp3 = 0.0

         do 1505 i1 = 1, ntstpt

*  FEEDFORWARD:  for each input node, compute an output on the first
*  layer x1(ii) and an input to the hidden layer.
*  use m+1 to include the bias term

         do 1513 ii=1,n
            b=0.
            do 1523 ll=1,m+1
1523             b=b+w1(ll,ii)*arrtst(ll,i1)
1513             x1(ii)=f(b)

* debug loop
      if(idbg.eq.1)then
        write(2,*)'Hidden Layer Outputs'
        write(2,*)x1
      endif

* set up bias term for hidden layer

         x1(n+1)=1.0

* continuation of feedforward loop.  compute an output term zz(jj)

         do 1533 jj=1,o
```

```
            b=0.
            do 1543 ii=1,n+1
1543             b=b+w2(ii,jj)*x1(ii)
                zz(jj)=f(b)
                temp1 = temp1 + (abs(zz(jj) - nint(arrtst(m+2,i1))))
                temp2 = temp2 + (zz(jj) - nint(arrtst(m+2,i1)))**2
                if ((abs(zz(jj) - nint(arrtst(m+2,i1)))).gt.temp3) then
                   temp3 = abs(zz(jj) - nint(arrtst(m+2,i1)))

*  temp3a and temp3b track the coordinates of the exemplar
*  causing the largest absolute error

                   temp3a = arrtst(1,i1)
                   temp3b = arrtst(2,i1)
                else
                   continue
                endif

1533       continue

1505      continue

         write (14,*) iepoch, temp1
         write (15,*) iepoch, temp2
         write (16,*) iepoch, temp3, temp3a, temp3b
      else
         continue
      endif

****
****   END COMPUTING TOTAL ABSOLUTE ERROR ON TESTING SET
****   THIS LOOP ALSO COMPUTES ERROR SUM OF SQUARES
****   THIS LOOP ALSO COMPUTES LARGEST ABSOLUTE ERROR
****

****
****   BEGIN WEIGHT CHANGE CALCULATIONS
****

      temp10 = 0.0
      temp11 = 0.0
      temp12 = 0.0
         do 1100 i = 1, m + 1
            do 1101 j = 1, n
               temp10 = temp10 + abs(w1temp(i,j) - w1(i,j))
               temp11 = temp11 + (w1temp(i,j) - w1(i,j))**2
               temp12 = temp12 + abs((w1temp(i,j) - w1(i,j))/w1(i,j))
               w1temp(i,j) = w1(i,j)
1101        continue
1100     continue
         do 1102 i = 1, n + 1
```

```
            do 1103 j = 1, o
               temp10 = temp10 + abs(w2temp(i,j) - w2(i,j))
               temp11 = temp11 + (w2temp(i,j) - w2(i,j))**2
               temp12 = temp12 + abs((w2temp(i,j) - w2(i,j))/w1(i,j))
               w2temp(i,j) = w2(i,j)
1103        continue
1102     continue
         write (41,*) iepoch, temp10
         write (42,*) iepoch, temp11
         temp13 = temp12/numwts
         write (43,*) iepoch, temp13
         avgwts = temp10/numwts
         write (44,*) iepoch, avgwts

****
****   END WEIGHT CHANGE CALCULATIONS
****

********************************************************************
*
*
*  END STOPPING CRITERIA CALCULATIONS
*
*
********************************************************************

********************************************************************
*
*
*  BEGIN CHECKING IF STOPPING CRITERIA HAVE BEEN MET
*
*
********************************************************************

*  check for stopping criteria for limit on number of cycles reached

       if (istcyc(ncycle).eq.0) then
          if(iepoch.ge.istop) then
             write (2,*) ' '
             write (2,*) 'REASON FOR STOP IS NUMBER OF EPOCHS'
             write (2,*) 'IT IS EPOCH NUMBER  ', iepoch
             write (2,*) 'ERROR RATE ON VAL SET IS ', 1.0 - clasrat5
             write (10,*) 'REASON FOR STOP IS NUMBER OF EPOCHS'
             write (10,*) 'IT IS EPOCH NUMBER  ', iepoch
             write (10,*) 'ERROR RATE ON VAL SET IS ', 1.0 - clasrat5
             write (50,*) iepoch, 1.-clasrat5, '  MAX NUM EPOCHS'
             istcyc(ncycle) = istcyc(ncycle) + 1
             iswitch = iswitch + 1
          else
             continue
          endif
```

```
        else
           continue
        endif

*   check for stopping criteria for low enough error rate on training set

       if (isttrn(ncycle).eq.0) then
          if(clasrat1.ge.errsttng) then
             write (2,*) ' '
             write (2,*) 'REASON FOR STOP IS ERROR RATE ON TRAINING SET'
             write (2,*) 'IT IS EPOCH NUMBER  ', iepoch
             write (2,*) 'ERROR RATE ON VAL SET IS ', 1.0 - clasrat5
             write(10,*) 'REASON FOR STOP IS ERROR RATE ON TRAINING SET'
             write (10,*) 'IT IS EPOCH NUMBER  ', iepoch
             write (10,*) 'ERROR RATE ON VAL SET IS ', 1.0 - clasrat5
             write (50,*) iepoch, 1.-clasrat5, '  ERR RATE ON TNG SET'
             isttrn(ncycle) = isttrn(ncycle) + 1
             iswitch = iswitch + 1
          else
             continue
          endif
        else
           continue
        endif

*   check for stopping criteria for low enough error rate on training set
*   occurring icont1 times in a row

       if (isttrnb(ncycle).eq.0) then

          if(clasrat1.ge.errsttng) then
              itempa(ncycle) = itempa(ncycle) + 1
          else
              itempa(ncycle) = 0
          endif

          if (itempa(ncycle).ge.icont1) then
             write (2,*) ' '
             write (2,*) 'REASON FOR STOP IS RPTD ERR RAT ON TNG SET'
             write (2,*) 'IT IS EPOCH NUMBER  ', iepoch
             write (2,*) 'ERROR RATE ON VAL SET IS ', 1.0 - clasrat5
             write(10,*) 'REASON FOR STOP IS RPTD ERR RAT ON TNG SET'
             write (10,*) 'IT IS EPOCH NUMBER  ', iepoch
             write (10,*) 'ERROR RATE ON VAL SET IS ', 1.0 - clasrat5
             write (50,*) iepoch, 1.-clasrat5, '  RPTD ER RAT TNG SET'
             isttrnb(ncycle) = isttrnb(ncycle) + 1
             iswitch = iswitch + 1
          else
             continue
          endif
        else
```

```
           continue
        endif

*  check for stopping criteria for low enough error rate on testing set

       if (isttst(ncycle).eq.0) then
          if(clasrat2.ge.errsttst) then
             write (2,*) ' '
             write (2,*) 'REASON FOR STOP IS ERROR RATE ON TESTING SET'
             write (2,*) 'IT IS EPOCH NUMBER  ', iepoch
             write (2,*) 'ERROR RATE ON VAL SET IS ', 1.0 - clasrat5
             write(10,*) 'REASON FOR STOP IS ERROR RATE ON TESTING SET'
             write (10,*) 'IT IS EPOCH NUMBER  ', iepoch
             write (10,*) 'ERROR RATE ON VAL SET IS ', 1.0 - clasrat5
             write (50,*) iepoch, 1.-clasrat5, '  ERR RATE ON TST SET'
             isttst(ncycle) = isttst(ncycle) + 1
             iswitch = iswitch + 1
          else
             continue
          endif
        else
           continue
        endif

*  check for stopping criteria for low enough error rate on testing set
*  occurring icont1 times in a row

       if (isttstb(ncycle).eq.0) then

          if(clasrat2.ge.errsttst) then
              itempb(ncycle) = itempb(ncycle) + 1
          else
              itempb(ncycle) = 0
          endif

          if (itempb(ncycle).ge.icont1) then
             write (2,*) ' '
             write (2,*) 'REASON FOR STOP IS RPTD ERR RAT ON TST SET'
             write (2,*) 'IT IS EPOCH NUMBER  ', iepoch
             write (2,*) 'ERROR RATE ON VAL SET IS ', 1.0 - clasrat5
             write(10,*) 'REASON FOR STOP IS RPTD ERR RAT ON TST SET'
             write (10,*) 'IT IS EPOCH NUMBER  ', iepoch
             write (10,*) 'ERROR RATE ON VAL SET IS ', 1.0 - clasrat5
             write (50,*) iepoch, 1.-clasrat5, '  RPTD ER RAT TST SET'
             isttstb(ncycle) = isttstb(ncycle) + 1
             iswitch = iswitch + 1
          else
             continue
          endif
        else
           continue
       endif

*  check for stopping criteria for low enough error rate on val set

      if (istval(ncycle).eq.0) then
         if(clasrat5.ge.errstval) then
            write (2,*) ' '
            write (2,*) 'REASON FOR STOP IS ERROR RATE ON VAL SET'
            write (2,*) 'IT IS EPOCH NUMBER  ', iepoch
            write (2,*) 'ERROR RATE ON VAL SET IS ', 1.0 - clasrat5
            write(10,*) 'REASON FOR STOP IS ERROR RATE ON VAL SET'
            write (10,*) 'IT IS EPOCH NUMBER  ', iepoch
            write (10,*) 'ERROR RATE ON VAL SET IS ', 1.0 - clasrat5
            write (50,*) iepoch, 1.-clasrat5, '  ERR RATE ON VAL SET'
            istval(ncycle) = istval(ncycle) + 1
            iswitch = iswitch + 1
         else
            continue
         endif
       else
          continue
       endif

*  check for stopping criteria for low weighted average error rate

      wtavg = pertrn*clasrat1 + pertst*clasrat2 + perval*clasrat5
      write(17,*) iepoch, 1.0 - wtavg

      if (istwa(ncycle).eq.0) then
         if(wtavg.ge.errstop2) then
            write (2,*) ' '
            write (2,*) 'REASON FOR STOP IS WEIGHT AVE ERROR RATE'
            write (2,*) 'IT IS EPOCH NUMBER  ', iepoch
            write (2,*) 'ERROR RATE ON VAL SET IS ', 1.0 - clasrat5
            write(10,*) 'REASON FOR STOP IS WEIGHT AVE ERROR RATE'
            write (10,*) 'IT IS EPOCH NUMBER  ', iepoch
            write (10,*) 'ERROR RATE ON VAL SET IS ', 1.0 - clasrat5
            write (50,*) iepoch, 1.-clasrat5, '  WT AVG ERR RATE '
            istwa(ncycle) = istwa(ncycle) + 1
            iswitch = iswitch + 1
         else
            continue
         endif
       else
          continue
       endif

*  check for stopping criteria for low weighted average error rate
*  occurring icont1 times in a row

      if (istwab(ncycle).eq.0) then
```

```
          if(wtavg.ge.errstop2) then
              itempc(ncycle) = itempc(ncycle) + 1
          else
              itempc(ncycle) = 0
          endif

          if (itempc(ncycle).ge.icont1) then
             write (2,*) ' '
             write (2,*) 'REASON FOR STOP IS RPTD WTD AVG ERROR RATE'
             write (2,*) 'IT IS EPOCH NUMBER  ', iepoch
             write (2,*) 'ERROR RATE ON VAL SET IS ', 1.0 - clasrat5
             write(10,*) 'REASON FOR STOP IS RPTD WTD AVG ERROR RATE'
             write (10,*) 'IT IS EPOCH NUMBER  ', iepoch
             write (10,*) 'ERROR RATE ON VAL SET IS ', 1.0 - clasrat5
             write (50,*) iepoch, 1.-clasrat5, '  RPTD WT AVG ERR RAT'
             istwab(ncycle) = istwab(ncycle) + 1
             iswitch = iswitch + 1
          else
             continue
          endif
        else
           continue
        endif

*  check for stopping criteria for total absolute error

       if (isttae(ncycle).eq.0) then
          if(temp1.le.abserrst) then
             write (2,*) ' '
             write (2,*) 'REASON FOR STOP IS TOTAL ABSOLUTE ERROR'
             write (2,*) 'IT IS EPOCH NUMBER  ', iepoch
             write (2,*) 'ERROR RATE ON VAL SET IS ', 1.0 - clasrat5
             write(10,*) 'REASON FOR STOP IS TOTAL ABSOLUTE ERROR'
             write (10,*) 'IT IS EPOCH NUMBER  ', iepoch
             write (10,*) 'ERROR RATE ON VAL SET IS ', 1.0 - clasrat5
             write (50,*) iepoch, 1.-clasrat5, '  TOT ABSOLUTE ERROR'
             isttae(ncycle) = isttae(ncycle) + 1
             iswitch = iswitch + 1
          else
             continue
          endif
        else
           continue
        endif

*  check for stopping criteria for total absolute error
*  occurring icont1 times in a row

       if (isttaeb(ncycle).eq.0) then
```

```
          if(temp1.le.abserrst) then
              itemp1(ncycle) = itemp1(ncycle) + 1
          else
              itemp1(ncycle) = 0
          endif

          if (itemp1(ncycle).ge.icont1) then
             write (2,*) ' '
             write (2,*) 'REASON FOR STOP IS REPEATED TOT ABS ERR'
             write (2,*) 'IT IS EPOCH NUMBER  ', iepoch
             write (2,*) 'ERROR RATE ON VAL SET IS ', 1.0 - clasrat5
             write(10,*) 'REASON FOR STOP IS REPEATED TOT ABS ERR'
             write (10,*) 'IT IS EPOCH NUMBER  ', iepoch
             write (10,*) 'ERROR RATE ON VAL SET IS ', 1.0 - clasrat5
             write (50,*) iepoch, 1.-clasrat5, '  RPTD TOT ABS ERROR'
             isttaeb(ncycle) = isttaeb(ncycle) + 1
             iswitch = iswitch + 1
          else
             continue
          endif
        else
           continue
        endif


*  check for stopping criteria for error sum of squares

       if (istess(ncycle).eq.0) then
          if(temp2.le.essstop1) then
             write (2,*) ' '
             write (2,*) 'REASON FOR STOP IS ERROR SUM OF SQUARES'
             write (2,*) 'IT IS EPOCH NUMBER  ', iepoch
             write (2,*) 'ERROR RATE ON VAL SET IS ', 1.0 - clasrat5
             write(10,*) 'REASON FOR STOP IS ERROR SUM OF SQUARES'
             write (10,*) 'IT IS EPOCH NUMBER  ', iepoch
             write (10,*) 'ERROR RATE ON VAL SET IS ', 1.0 - clasrat5
             write (50,*) iepoch, 1.-clasrat5, '  ERR SUM OF SQUARES'
             istess(ncycle) = istess(ncycle) + 1
             iswitch = iswitch + 1
          else
             continue
          endif
        else
           continue
        endif

*  check for stopping criteria for error sum of squares
*  occurring icont1 times in a row

       if (istessb(ncycle).eq.0) then
```

```
          if(temp2.le.essstop1) then
              itemp2(ncycle) = itemp2(ncycle) + 1
          else
              itemp2(ncycle) = 0
          endif

          if (itemp2(ncycle).ge.icont1) then
             write (2,*) ' '
             write (2,*) 'REASON FOR STOP IS REPEATED ERR SUM SQ'
             write (2,*) 'IT IS EPOCH NUMBER  ', iepoch
             write (2,*) 'ERROR RATE ON VAL SET IS ', 1.0 - clasrat5
             write(10,*) 'REASON FOR STOP IS REPEATED ERR SUM SQ'
             write (10,*) 'IT IS EPOCH NUMBER  ', iepoch
             write (10,*) 'ERROR RATE ON VAL SET IS ', 1.0 - clasrat5
             write (50,*) iepoch, 1.-clasrat5, '  RPTD ERR SUM OF SQRS'
             istessb(ncycle) = istessb(ncycle) + 1
             iswitch = iswitch + 1
          else
             continue
          endif
        else
           continue
        endif

*  check for stopping criteria for largest absolute error

       if (istlae(ncycle).eq.0) then
          if(temp3.le.errlrg) then
             write (2,*) ' '
             write (2,*) 'REASON FOR STOP IS LARGEST ABSOLT ERROR'
             write (2,*) 'IT IS EPOCH NUMBER  ', iepoch
             write (2,*) 'ERROR RATE ON VAL SET IS ', 1.0 - clasrat5
             write(10,*) 'REASON FOR STOP IS LARGEST ABSOLT ERROR'
             write (10,*) 'IT IS EPOCH NUMBER  ', iepoch
             write (10,*) 'ERROR RATE ON VAL SET IS ', 1.0 - clasrat5
             write (50,*) iepoch, 1.-clasrat5, '  LARGEST ABS ERROR'
             istlae(ncycle) = istlae(ncycle) + 1
             iswitch = iswitch + 1
          else
             continue
          endif
        else
           continue
        endif

*  check for stopping criteria for largest absolute error
*  occurring icont1 times in a row

       if (istlaeb(ncycle).eq.0) then

          if(temp3.le.errlrg) then
```

```
              itemp3(ncycle) = itemp3(ncycle) + 1
          else
              itemp3(ncycle) = 0
          endif

          if (itemp3(ncycle).ge.icont1) then
             write (2,*) ' '
             write (2,*) 'REASON FOR STOP IS RPEATED LRG ABS ERR'
             write (2,*) 'IT IS EPOCH NUMBER  ', iepoch
             write (2,*) 'ERROR RATE ON VAL SET IS ', 1.0 - clasrat5
             write(10,*) 'REASON FOR STOP IS RPEATED LRG ABS ERR'
             write (10,*) 'IT IS EPOCH NUMBER  ', iepoch
             write (10,*) 'ERROR RATE ON VAL SET IS ', 1.0 - clasrat5
             write (50,*) iepoch, 1.-clasrat5, '  RPTD LARGE ABS ERROR'
             istlaeb(ncycle) = istlaeb(ncycle) + 1
             iswitch = iswitch + 1
          else
             continue
          endif
        else
           continue
        endif

*  check for stopping criteria for absolute weight change

       if (istawc(ncycle).eq.0) then
          if(temp10.le.awcstop) then
             write (2,*) ' '
             write (2,*) 'REASON FOR STOP IS ABSOLUTE WEIGHT CHANGE'
             write (2,*) 'IT IS EPOCH NUMBER  ', iepoch
             write (2,*) 'ERROR RATE ON VAL SET IS ', 1.0 - clasrat5
             write(10,*) 'REASON FOR STOP IS ABSOLUTE WEIGHT CHANGE'
             write (10,*) 'IT IS EPOCH NUMBER  ', iepoch
             write (10,*) 'ERROR RATE ON VAL SET IS ', 1.0 - clasrat5
             write (50,*) iepoch, 1.-clasrat5, '  ABSOLUTE WEIGHT CHG'
             istawc(ncycle) = istawc(ncycle) + 1
             iswitch = iswitch + 1
          else
             continue
          endif
        else
           continue
        endif

*  check for stopping criteria for squared weight change

       if (istswc(ncycle).eq.0) then
          if(temp11.le.swcstop) then
             write (2,*) ' '
             write (2,*) 'REASON FOR STOP IS SQUARED WEIGHT CHANGE'
             write (2,*) 'IT IS EPOCH NUMBER  ', iepoch
             write (2,*) 'ERROR RATE ON VAL SET IS ', 1.0 - clasrat5
             write(10,*) 'REASON FOR STOP IS SQUARED WEIGHT CHANGE'
             write (10,*) 'IT IS EPOCH NUMBER  ', iepoch
             write (10,*) 'ERROR RATE ON VAL SET IS ', 1.0 - clasrat5
             write (50,*) iepoch, 1.-clasrat5, '  SQUARED WEIGHT CHG'
             istswc(ncycle) = istswc(ncycle) + 1
             iswitch = iswitch + 1
          else
             continue
          endif
        else
           continue
        endif

*  check for stopping criteria for absolute relative weight change

       if (istrwc(ncycle).eq.0) then
          if(temp13.le.rwcstop) then
             write (2,*) ' '
             write (2,*) 'REASON FOR STOP IS RELATIVE WEIGHT CHANGE'
             write (2,*) 'IT IS EPOCH NUMBER  ', iepoch
             write (2,*) 'ERROR RATE ON VAL SET IS ', 1.0 - clasrat5
             write(10,*) 'REASON FOR STOP IS RELATIVE WEIGHT CHANGE'
             write (10,*) 'IT IS EPOCH NUMBER  ', iepoch
             write (10,*) 'ERROR RATE ON VAL SET IS ', 1.0 - clasrat5
             write (50,*) iepoch, 1.-clasrat5, '  RELATIVE WEIGHT CHG'
             istrwc(ncycle) = istrwc(ncycle) + 1
             iswitch = iswitch + 1
          else
             continue
          endif
        else
           continue
        endif

*  check for stopping criteria for absolute relative weight change
*  occurring icont1 times in a row

       if (istrwcb(ncycle).eq.0) then

          if(temp13.le.rwcstop) then
              itempf(ncycle) = itempf(ncycle) + 1
          else
              itempf(ncycle) = 0
          endif

          if (itempf(ncycle).ge.icont1) then
             write (2,*) ' '
             write (2,*) 'REASON FOR STOP IS REPEATED REL WT CHG'
             write (2,*) 'IT IS EPOCH NUMBER  ', iepoch
             write (2,*) 'ERROR RATE ON VAL SET IS ', 1.0 - clasrat5
             write(10,*) 'REASON FOR STOP IS REPEATED REL WT CHG'
             write (10,*) 'IT IS EPOCH NUMBER  ', iepoch
             write (10,*) 'ERROR RATE ON VAL SET IS ', 1.0 - clasrat5
             write (50,*) iepoch, 1.-clasrat5, '  RPTD REL WT CHG'
             istrwcb(ncycle) = istrwcb(ncycle) + 1
             iswitch = iswitch + 1
          else
             continue
          endif
        else
           continue
        endif

*  check for stopping criteria for absolute mean weight change

       if (istmwc(ncycle).eq.0) then
          if(avgwts.le.mwcstop) then
             write (2,*) ' '
             write (2,*) 'REASON FOR STOP IS MEAN WEIGHT CHANGE'
             write (2,*) 'IT IS EPOCH NUMBER  ', iepoch
             write (2,*) 'ERROR RATE ON VAL SET IS ', 1.0 - clasrat5
             write(10,*) 'REASON FOR STOP IS MEAN WEIGHT CHANGE'
             write (10,*) 'IT IS EPOCH NUMBER  ', iepoch
             write (10,*) 'ERROR RATE ON VAL SET IS ', 1.0 - clasrat5
             write (50,*) iepoch, 1.-clasrat5, '  MEAN WEIGHT CHG'
             istmwc(ncycle) = istmwc(ncycle) + 1
             iswitch = iswitch + 1
          else
             continue
          endif
        else
           continue
        endif

*  check for stopping criteria for moving average error
*  need more than five epochs

      if (iepoch.gt.5) then

       if (istma(ncycle).eq.0) then
          if(movavg(intervl).le.mastop) then
             write (2,*) ' '
             write (2,*) 'REASON FOR STOP IS MOVING AVERAGE ERROR'
             write (2,*) 'IT IS EPOCH NUMBER  ', iepoch
             write (2,*) 'ERROR RATE ON VAL SET IS ', 1.0 - clasrat5
             write(10,*) 'REASON FOR STOP IS MOVING AVERAGE ERROR'
             write (10,*) 'IT IS EPOCH NUMBER  ', iepoch
             write (10,*) 'ERROR RATE ON VAL SET IS ', 1.0 - clasrat5
             write (50,*) iepoch, 1.-clasrat5, '  MOVING AVG ERROR'
             istma(ncycle) = istma(ncycle) + 1
             iswitch = iswitch + 1
          else
            continue
          endif
        else
           continue
        endif

       else
          continue
       endif

*  BEGIN A MAP FOR EACH 50 EPOCHS

      if ( (iepoch.eq.50).or.(iepoch.eq.100).or.(iepoch.eq.150)
     *   .or.
     *   (iepoch.eq.200).or.(iepoch.eq.250).or.(iepoch.eq.300)
     *   .or.
     *   (iepoch.eq.350).or.(iepoch.eq.400).or.(iepoch.eq.450)
     *   .or. (iepoch.eq.500) ) then

            write (2,*) ' '
            write (2,*) 'PERIODIC EPOCH REPORT'
            write (2,*) 'IT IS EPOCH NUMBER  ', iepoch
            write (2,*) 'ERROR RATE ON VAL SET IS ', 1.0 - clasrat5
            write(10,*) 'PERIODIC EPOCH REPORT'
            write (10,*) 'IT IS EPOCH NUMBER  ', iepoch
            write (10,*) 'ERROR RATE ON VAL SET IS ', 1.0 - clasrat5
            write (50,*) iepoch, 1.-clasrat5, ' PERIOD EPOCH REPORT'
            iswitch = iswitch + 1

       else
          continue
       endif

*********************************************************************
*
*
*  BEGIN TEST FOR GRAPH DATA FOR TOTAL ABSOLUTE ERROR
*  This section runs at the end of each epoch to determine if a
*       threshold has been crossed.  Initialize igrafXX at the
*       beginning of the program and write to file fort.XX
*       the first time the threshold is met
*  Use igrafXX(ncycle) as a switch to see if this threshold
*      has been met before
*  Write to files number 61 and higher
*  temp1 is the total absolute error value
*
*
*********************************************************************

       if (igraf01(ncycle).eq.0) then
```

```
         if (temp1.le.(1.0)) then
            write(61,*) ncycle, iepoch, 1.0 - clasrat5
            igraf01(ncycle) = igraf01(ncycle) + 1
         else
            continue
         endif
      else
         continue
      endif

      if (igraf02(ncycle).eq.0) then
         if (temp1.le.(2.0)) then
            write(62,*) ncycle, iepoch, 1.0 - clasrat5
            igraf02(ncycle) = igraf02(ncycle) + 1
         else
            continue
         endif
      else
         continue
      endif

**********************************************************************
*
*
*  END TEST FOR GRAPH DATA FOR TOTAL ABSOLUTE ERROR
*
*
**********************************************************************

*  iswitch is greater than 0 if a tolerance threshold had been met.

      if (iswitch.gt.0) then
         go to 3
      else
         continue
      endif

**********************************************************************
*
*
*  END CHECKING IF STOPPING CRITERIA HAVE BEEN MET
*
*
**********************************************************************

*  Take the go to 1 loop if program doesn't go to 3 above

      go to 1

* take else route if the entire training set isn't complete this epoch.
*  This goes far back into the program to label 99.
```

```
      else
        go to 2
      endif

*  When an iswitch value indicates that one of the potential stopping
*  criterion has indicated a stopping point or a periodic activation
*  map is desired, enter here to print out results to fort.2 and fort.10

3     if(iprint.eq.1)then

       rewind 30

       x0(3) = 1.0

       do 5000 i=11,1,-1
        do 5000 j=1,11

           read(30,*) x0(1), x0(2)

C FEEDFORWARD

          do 210 ii=1,n
            a=0.
            do 220 ll=1,m+1
220           a=a+w1(ll,ii)*x0(ll)
210         x1(ii)=f(a)

          x1(n+1)=1.0
          do 230 jj=1,o
            a=0.
            do 240 ii=1,n+1
240           a=a+w2(ii,jj)*x1(ii)
230         zz(jj)=f(a)
          xx(i,j)=ifix(100.*zz(1))
5000   continue

*  Write to the activation map in fort.10

      write (10,*) ' '
      write (10,*) 'ACTIVATION MAP FOR CYCLE NUMBER  ', ncycle
      write (10,*) ' '

      do 5001 i=11,1,-1
         write(10,1111)(xx(i,j),j=1,11)
1111     format(11(1x,i3))
5001   continue

      write (10,*) ' '
      write (10,*) ' '
```

```
* Write final weights to file fort.2

      write(2,*)'FINAL LAYER 1 WEIGHTS '
      do 410 i=1,m+1
410     write(2,131) (w1(i,j),j=1,n)
      write(2,*)'FINAL LAYER 2 WEIGHTS'
      do 411 i=1,n+1
411     write(2,131)(w2(i,j),j=1,o)

       endif

************************************************************************
*
*  Determine if all of the possible stopping criteria have indicated
*  that it is time to quit this cycle.
*
************************************************************************

      if ((istcyc(ncycle).eq.1).and.(isttrn(ncycle).eq.1)
     *     .and.(isttst(ncycle).eq.1).and.(istval(ncycle).eq.1)
     *     .and.(istwa(ncycle).eq.1).and.(isttae(ncycle).eq.1)
     *     .and.(isttaeb(ncycle).eq.1).and.(istess(ncycle).eq.1)
     *     .and.(istessb(ncycle).eq.1).and.(istlae(ncycle).eq.1)
     *     .and.(istlaeb(ncycle).eq.1)) then
           goto 507
      else
           if (istcyc(ncycle).eq.1) then
              if (isttrn(ncycle).eq.0) then
                write (2,*) ' '
                write (2,*) 'TRAIN SET NOT MEET ERR CRIT IN EPOCHS'
                write (10,*) ' '
                write (10,*) 'TRAIN SET NOT MEET ERR CRIT IN EPOCHS'
              else
                continue
              endif
           endif

           if (istcyc(ncycle).eq.1) then
              if (isttrnb(ncycle).eq.0) then
                write (2,*) ' '
                write (2,*) 'RPTD TRN SET NOT MEET ERR CRIT IN EPOCHS'
                write (10,*) ' '
                write (10,*) 'RPTD TRN SET NOT MEET ERR CRIT IN EPCHS'
              else
                continue
              endif
           endif

           if (istcyc(ncycle).eq.1) then
              if (isttst(ncycle).eq.0) then
                write (2,*) ' '
     write (2,*) 'TEST SET NOT MEET ERR CRIT IN EPOCHS'
     write (10,*) ' '
     write (10,*) 'TEST SET NOT MEET ERR CRIT IN EPOCHS'
   else
     continue
   endif
endif

if (istcyc(ncycle).eq.1) then
   if (isttstb(ncycle).eq.0) then
     write (2,*) ' '
     write (2,*) 'RPTD TST SET NOT MEET ERR CRIT IN EPOCHS'
     write (10,*) ' '
     write (10,*) 'RPTD TST SET NOT MEET ERR CRIT IN EPCHS'
   else
     continue
   endif
endif


if (istcyc(ncycle).eq.1) then
   if (istval(ncycle).eq.0) then
     write (2,*) ' '
     write (2,*) 'VAL SET NOT MEET ERR CRIT IN EPOCHS'
     write (10,*) ' '
     write (10,*) 'VAL SET NOT MEET ERR CRIT IN EPOCHS'
   else
     continue
   endif
endif

if (istcyc(ncycle).eq.1) then
   if (istwa(ncycle).eq.0) then
     write (2,*) ' '
     write (2,*) 'WEIGHT AVE NOT MEET ERR CRIT IN EPOCHS'
     write (10,*) ' '
     write (10,*) 'WEIGHT AVE NOT MEET ERR CRIT IN EPOCHS'
   else
     continue
   endif
endif

if (istcyc(ncycle).eq.1) then
   if (istwab(ncycle).eq.0) then
     write (2,*) ' '
     write (2,*) 'RPTD WT AVG NOT MEET ERR CRIT IN EPOCHS'
     write (10,*) ' '
     write (10,*) 'RPT WT AVG NOT MEET ERR CRIT IN EPOCHS'
   else
     continue
   endif
```

```
endif

if (istcyc(ncycle).eq.1) then
   if (isttae(ncycle).eq.0) then
     write (2,*) ' '
     write (2,*) 'TOT ABS ERR NOT MEET ERR CRIT IN EPOCHS'
     write (10,*) ' '
     write (10,*) 'TOT ABS ERR NOT MEET ERR CRIT IN EPCHS'
   else
     continue
   endif
endif

if (istcyc(ncycle).eq.1) then
   if (isttaeb(ncycle).eq.0) then
     write (2,*) ' '
     write (2,*) 'RPTED TOT ABS ERR NOT MET ER CRT IN EP'
     write (10,*) ' '
     write (10,*)'RPTED TOT ABS ERR NOT MET ER CRT IN EP'
   else
     continue
   endif
endif

if (istcyc(ncycle).eq.1) then
   if (istess(ncycle).eq.0) then
     write (2,*) ' '
     write (2,*) 'ERR SUM SQ NOT MEET ERR CRIT IN EPOCHS'
     write (10,*) ' '
     write (10,*) 'ERR SUM SQ NOT MEET ERR CRIT IN EPCHS'
   else
     continue
   endif
endif

if (istcyc(ncycle).eq.1) then
   if (istessb(ncycle).eq.0) then
     write (2,*) ' '
     write (2,*) 'RPTED ERR SUM SQ NOT MET ER CRT IN EP'
     write (10,*) ' '
     write (10,*)'RPTED ERR SUM SQ NOT MET ER CRT IN EP'
   else
     continue
   endif
endif

if (istcyc(ncycle).eq.1) then
   if (istlae(ncycle).eq.0) then
     write (2,*) ' '
     write (2,*) 'LRG ABS ER NOT MEET ERR CRIT IN EPOCHS'
     write (10,*) ' '
     write (10,*) 'LRG ABS ER NOT MEET ERR CRIT IN EPCHS'
   else
     continue
   endif
endif

if (istcyc(ncycle).eq.1) then
   if (istlaeb(ncycle).eq.0) then
     write (2,*) ' '
     write (2,*) 'RPTED LRG ABS ER NOT MET ER CRT IN EP'
     write (10,*) ' '
     write (10,*)'RPTED LRG ABS ER NOT MET ER CRT IN EP'
   else
     continue
   endif
endif

if (istcyc(ncycle).eq.1) then
   if (istawc(ncycle).eq.0) then
     write (2,*) ' '
     write (2,*) 'ABS WGHT CHANGE NOT MET ER CRT IN EP'
     write (10,*) ' '
     write (10,*)'ABS WGHT CHANGE NOT MET ER CRT IN EP'
   else
     continue
   endif
endif

if (istcyc(ncycle).eq.1) then
   if (istswc(ncycle).eq.0) then
     write (2,*) ' '
     write (2,*) 'SQR WGHT CHANGE NOT MET ER CRT IN EP'
     write (10,*) ' '
     write (10,*)'SQR WGHT CHANGE NOT MET ER CRT IN EP'
   else
     continue
   endif
endif

if (istcyc(ncycle).eq.1) then
   if (istrwc(ncycle).eq.0) then
     write (2,*) ' '
     write (2,*) 'REL WGHT CHANGE NOT MET ER CRT IN EP'
     write (10,*) ' '
     write (10,*)'REL WGHT CHANGE NOT MET ER CRT IN EP'
   else
     continue
   endif
endif

if (istcyc(ncycle).eq.1) then
```

```
              if (istrwcb(ncycle).eq.0) then
                write (2,*) ' '
                write (2,*) 'RPTD REL WT CHG NOT MET ER CRT IN EP'
                write (10,*) ' '
                write (10,*)'RPTD REL WT CHG NOT MET ER CRT IN EP'
              else
                continue
              endif
           endif

           if (istcyc(ncycle).eq.1) then
              if (istmwc(ncycle).eq.0) then
                write (2,*) ' '
                write (2,*) 'MEAN WGHT CHANGE NOT MET ER CRT IN EP'
                write (10,*) ' '
                write (10,*)'MEAN WGHT CHANGE NOT MET ER CRT IN EP'
              else
                continue
              endif
           endif

           if (istcyc(ncycle).eq.1) then
              if (istma(ncycle).eq.0) then
                write (2,*) ' '
                write (2,*) 'MOVING AVG ERROR NOT MET ER CRT IN EP'
                write (10,*) ' '
                write (10,*)'MOVING AVG ERROR NOT MET ER CRT IN EP'
              else
                continue
              endif
           endif

           if (istcyc(ncycle).eq.1) then
              goto 507
           else
              continue
           endif

           goto 1
      endif

*  Update the number of cycles

507      ncycle = ncycle + 1
         write (50,*) ' '
         write (50,*) 'STARTING CYCLE NUMBER ', ncycle
         write (50,*) ' '

*  Determine if there are a sufficient number of cycles.

      if (ncycle.le.ncstop) then
```

```
          goto 501
      else
          continue
      endif

      stop
      end

      real function f(x)
      f=1./(1.+exp(-1.*x))
      return
      end

******************************************************************************
*  RANDOM NUMBER GENERATOR
*
*  Returns a uniform random deviate between 0.0 and 1.0 using a system
*  Set IDUM to any negative value to initialize or reinitialize the sequence.
*
*  Reference:  Numerical Recipes, p. 196
*
******************************************************************************

       FUNCTION RAN1(IDUM)

       DIMENSION R(97)
       PARAMETER (M1=259200,IA1=7141,IC1=54773,RM1=1./M1)
       PARAMETER (M2=134456,IA2=8121,IC2=28411,RM2=1./M2)
       PARAMETER (M3=243000,IA3=4561,IC3=51349)
       DATA IFF /0/
       IF (IDUM.LT.0.OR.IFF.EQ.0) THEN
         IFF=1
         IX1=MOD(IC1-IDUM,M1)
         IX1=MOD(IA1*IX1+IC1,M1)
         IX2=MOD(IX1,M2)
         IX1=MOD(IA1*IX1+IC1,M1)
         IX3=MOD(IX1,M3)
         DO 11 J=1,97
           IX1=MOD(IA1*IX1+IC1,M1)
           IX2=MOD(IA2*IX2+IC2,M2)
           R(J)=(FLOAT(IX1)+FLOAT(IX2)*RM2)*RM1
 11      CONTINUE
         IDUM=1
       ENDIF
       IX1=MOD(IA1*IX1+IC1,M1)
       IX2=MOD(IA2*IX2+IC2,M2)
       IX3=MOD(IA3*IX3+IC3,M3)
       J=1+(97*IX3)/M3
       IF(J.GT.97.OR.J.LT.1)PAUSE
       RAN1=R(J)
       R(J)=(FLOAT(IX1)+FLOAT(IX2)*RM2)*RM1
      RETURN
      END

*******************************************************************************
*  RANDOM NUMBER GENERATOR
*
*  Returns a uniform random deviate between 0.0 and 1.0 using a system
*  Set IDUM2 to any negative value to initialize or reinitialize the sequence.
*
*  Reference:  Numerical Recipes, p. 196
*
*******************************************************************************

      FUNCTION RAN2(IDUM2)

      DIMENSION R(97)
      PARAMETER (M1=259200,IA1=7141,IC1=54773,RM1=1./M1)
      PARAMETER (M2=134456,IA2=8121,IC2=28411,RM2=1./M2)
      PARAMETER (M3=243000,IA3=4561,IC3=51349)
      DATA IFF /0/
      IF (IDUM2.LT.0.OR.IFF.EQ.0) THEN
        IFF=1
        IX1=MOD(IC1-IDUM2,M1)
        IX1=MOD(IA1*IX1+IC1,M1)
        IX2=MOD(IX1,M2)
        IX1=MOD(IA1*IX1+IC1,M1)
        IX3=MOD(IX1,M3)
        DO 11 J=1,97
          IX1=MOD(IA1*IX1+IC1,M1)
          IX2=MOD(IA2*IX2+IC2,M2)
          R(J)=(FLOAT(IX1)+FLOAT(IX2)*RM2)*RM1
 11     CONTINUE
        IDUM2=1
      ENDIF
      IX1=MOD(IA1*IX1+IC1,M1)
      IX2=MOD(IA2*IX2+IC2,M2)
      IX3=MOD(IA3*IX3+IC3,M3)
      J=1+(97*IX3)/M3
      IF(J.GT.97.OR.J.LT.1)PAUSE
      RAN2=R(J)
      R(J)=(FLOAT(IX1)+FLOAT(IX2)*RM2)*RM1
      RETURN
      END

**********************************************************************
*  RANDOM NUMBER GENERATOR.  RANO
*
*  Written by Lisa Belue.
*
*  Returns a uniform random deviate between 0.0 and 1.0 using a system
*  supplied routine RAN(ISEED).  Set IDUM to any negative value to
*  initialize or reinitialize the sequence.
*
**********************************************************************

       FUNCTION RAN0(IDUM0)

       DIMENSION V(97)
       DATA IFF /0/
       IF (IDUM.LT.0.OR.IFF.EQ.0) THEN
         IFF=1
         ISEED=ABS(IDUM)
         IDUM=1
         DO 11 J=1,97
           DUM=RAN(ISEED)
 11      CONTINUE
         DO 12 J=1,97
           V(J)=RAN(ISEED)
 12      CONTINUE
         Y=RAN(ISEED)
      ENDIF
      J=1+INT(97.*Y)
      IF(J.GT.97.OR.J.LT.1) then
        PAUSE
      else
        continue
      endif
      Y=V(J)
      RAN0=Y
      V(J)=RAN(ISEED)
      RETURN
      END
```

Appendix B. Stopping Criteria Formulations

This section lists the mathematical formulation of the stopping criteria evaluated in the artificial neural network (ANN).

- **Classification Error Rate**

  The classification error rate is evaluated for the training and testing sets and is defined as follows:

  $$R = \frac{c}{n}$$

  where c is the number of exemplars classified correctly by the ANN and n is the total number of exemplars classified.

- **Weighted Average Error Rate**

  Weighted average error rate is evaluated using both the training set and the testing set classification error rates as determined above. It is defined as follows:

  $$WtAvg = p_{train} * R_{train} + p_{test} * R_{test}$$

  where R_{train} is the classification error rate for the training set, R_{test} is the classification error rate for the testing set, p_{train} is the percentage weight assigned to the training set, and p_{test} is the percentage weight assigned to the testing set such that

  $$p_{train} + p_{test} = 1$$

- **Moving Average Error Rate**

  Moving average error rate is evaluated using the testing set over the last *Int* epochs and is defined as follows:

$$MovAvg = \sum_{i}^{Int} \frac{R_i}{Int}$$

  where R_i is the classification error rate for the testing set.

- **Total Absolute Error**

  Total absolute error is evaluated on the testing set and is defined as follows:

$$E = \sum_{i}^{n} |y_i - d_i|$$

  where y_i is the ANN output for exemplar i, d_i is the known, desired output for exemplar i, and n is the number of testing exemplars.

- **Error Sum of Squares**

  Error sum of squares is evaluated on the testing set and is defined as follows:

$$ESS = \sum_{i}(y_i - d_i)^2$$

  where y_i is the ANN output for exemplar i, d_i is the known, desired output for exemplar i, and n is the number of testing exemplars.

- **Largest Absolute Error**

  Largest absolute error is evaluated on the testing set and is defined as follows:

$$LAE = max|y_i - d_i|$$

  where y_i is the ANN output for exemplar i, d_i is the known, desired output for exemplar i, and n is the number of testing exemplars.

- **Absolute Weight Change**

  Absolute weight change is defined as follows:

  $$AWC = \sum_{k=1}^{m} |w_k(t+1) - w_k(t)|$$

  where m is the total number of weights and w_k is the weight of neuron k for the tth epoch.

- **Squared Weight Change**

  Squared weight change is defined as follows:

  $$SWC = \sum_{k=1}^{m} [w_k(t+1) - w_k(t)]^2$$

  where m is the total number of weights and w_k is the weight of neuron k for the tth epoch.

- **Relative Weight Change**

  Relative weight change is defined as follows:

  $$RWC = \sum_{k=1}^{m} \frac{|w_k(t+1) - w_k(t)|}{|w_k(t)|}$$

  where m is the total number of weights and w_k is the weight of neuron k for the tth epoch.

- **Mean Weight Change**

  Mean weight change is defined as follows:

  $$MWC = \frac{\sum_{k=1}^{m} |w_k(t+1) - w_k(t)|}{m}$$

  where m is the total number of weights and w_k is the weight of neuron k for the tth epoch.

Appendix C. Total Absolute Error Function in the XOR Problem

This appendix shows the function of total absolute error for the four cases which did not meet the thresholds of the third experiment. They are listed by the replication number (out of ten replications) in which they occurred.

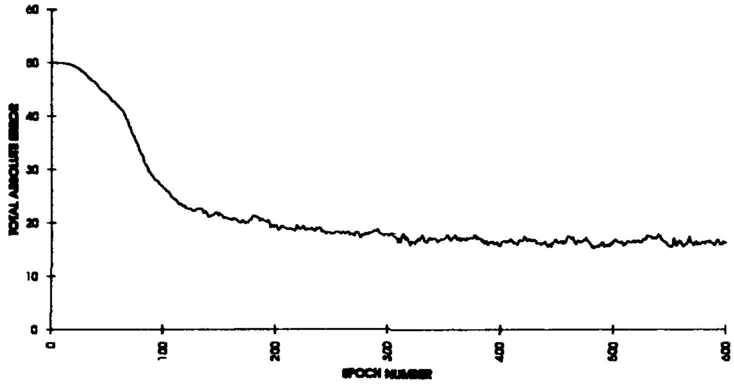


Figure 42. Replication One Total Absolute Error

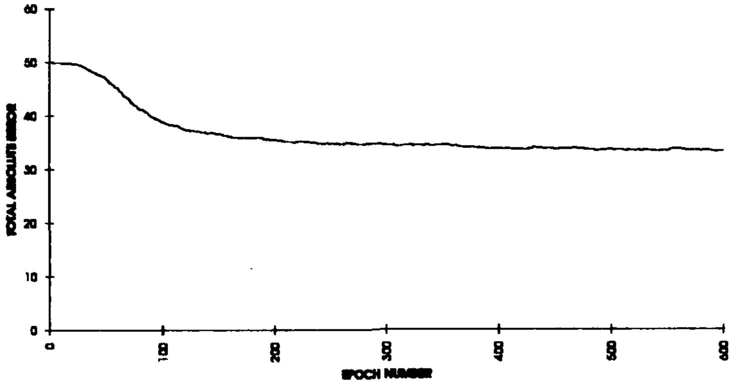


Figure 43. Replication Four Total Absolute Error

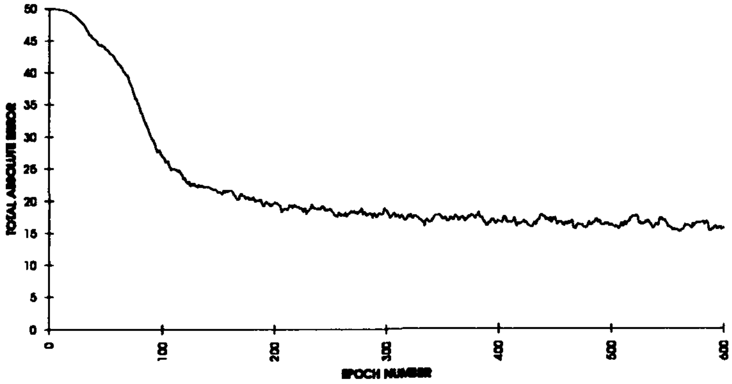


Figure 44. Replication Eight Total Absolute Error

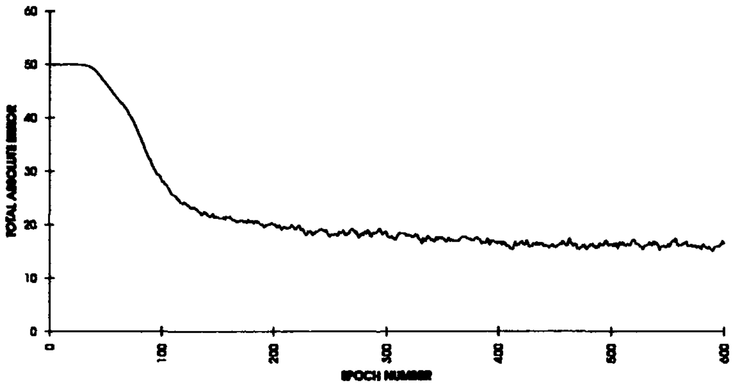


Figure 45. Replication Nine Total Absolute Error

Appendix D. Mesh Problem: An Output Comparison

This chapter contains the raw data for the mesh problem. The artificial neural network was trained for up to 15,000 epochs. The number of hidden nodes was set at all even numbers from four to twenty (inclusive). At each level of hidden nodes, ten replications of the training were conducted. In the Tables which follow, the asterisks (*) indicate those values selected by the heuristic rules given in Chapter 4.

Table 6: Four Hidden Node Output

Cycle	Statistic	Epoch	Statistic	Val Set Error Rate	Ave Error Rate or Std Dev	Select Ave Er Rat or St Dev
Training Set						
1*	0.1200	2901	0.1200	0.2183	0.2617	0.2294
2*	0.1100	13920	0.1100	0.2200		
3	0.1600	12727	0.1600	0.2633	0.0466	0.0142
4	0.2000	9462	0.2000	0.3250		
5*	0.1200	13633	0.1200	0.2200		
6	0.1900	14328	0.1900	0.3300		
7*	0.1100	12431	0.1100	0.2450		
8*	0.1200	11577	0.1200	0.2233		
9	0.1800	6552	0.1800	0.3217		
10*	0.1200	9652	0.1200	0.2500		
Testing Set						
1*	0.2300	3047	0.2300	0.2050	0.2532	0.2217
2*	0.2300	1091	0.2300	0.2067		
3*	0.2500	4632	0.2500	0.2667	0.0543	0.0236
4	0.3700	10121	0.3700	0.3233		
5*	0.2300	2697	0.2300	0.2000		
6	0.3700	10951	0.3700	0.3250		
7*	0.2400	995	0.2400	0.2317		
8*	0.2400	669	0.2400	0.2100		
9	0.3700	2614	0.3700	0.3317		
10*	0.2400	933	0.2400	0.2317		

Table 6: Four Hidden Node Output (continued)

Cycle	Statistic	Epoch	Statistic	Val Set Error Rate	Ave Error Rate or Std Dev	Select Ave Er Rat or St Dev
Weighted Average						
1*	0.1850	2901	0.1850	0.2183	0.2568	0.2204
2*	0.1850	1906	0.1850	0.2183		
3	0.2100	12727	0.2100	0.2633	0.0497	0.0025
4	0.2900	9462	0.2900	0.3250		
5	0.2050	6341	0.2050	0.2050		
6	0.2850	14536	0.2850	0.3283		
7*	0.1950	1188	0.1950	0.2217		
8*	0.1900	11577	0.1900	0.2233		
9	0.2850	6552	0.2850	0.3217		
10	0.2000	14949	0.2000	0.2433		
Moving Average						
1*	0.2400	3139	0.2400	0.2167	0.2553	0.2255
2*	0.2360	1101	0.2360	0.2100		
3*	0.2520	6132	0.2520	0.2633	0.0505	0.0187
4	0.3760	11693	0.3760	0.3200		
5*	0.2540	2700	0.2540	0.2083		
6	0.3740	13176	0.3740	0.3233		
7*	0.2500	1038	0.2500	0.2250		
8*	0.2400	13109	0.2400	0.2233		
9	0.3780	2614	0.3780	0.3317		
10*	0.2500	924	0.2500	0.2317		
Total Absolute Error						
1*	28.9528	6125	28.9528	0.2067	0.2578	0.2230
2*	29.4051	5349	29.4051	0.2067		
3*	28.2199	14052	28.2199	0.2683	0.0512	0.0263
4	42.8098	10456	42.8098	0.3250		
5*	29.3826	6335	29.3826	0.2100		
6	42.9163	12987	42.9163	0.3250		
7	30.7562	12501	30.7562	0.2417		
8*	26.6532	14685	26.6532	0.2233		
9	42.5929	14058	42.5929	0.3300		
10	30.7660	11150	30.7660	0.2417		

Table 7: Six Hidden Node Output

				Val Set	Ave Error	Select
				Error	Rate or	Ave Er Rat
Cycle	Statistic	Epoch	Statistic	Rate	Std Dev	or St Dev
Training Set						
1	0.1100	4268	0.1100	0.2167	0.1873	0.1588
2	0.1000	14096	0.1000	0.1983		
3	0.1000	12515	0.1000	0.1933	0.0301	0.0270
4*	0.0800	5381	0.0800	0.1733		
5	0.1000	14756	0.1000	0.2033		
6*	0.0700	10709	0.0700	0.1717		
7	0.1000	11960	0.1000	0.2067		
8*	0.0700	5688	0.0700	0.1183		
9	0.1200	2317	0.1200	0.2200		
10*	0.0800	6254	0.0800	0.1717		
Testing Set						
1	0.2100	10493	0.2100	0.1967	0.1760	0.1570
2	0.2100	6931	0.2100	0.1917		
3	0.2000	4302	0.2000	0.1933	0.0317	0.0367
4*	0.1900	1680	0.1900	0.1783		
5	0.2000	5321	0.2000	0.1917		
6*	0.1800	1606	0.1800	0.1600		
7	0.2400	3144	0.2400	0.2017		
8*	0.1000	2787	0.1000	0.0933		
9*	0.1800	12483	0.1800	0.1833		
10*	0.1900	3620	0.1900	0.1700		
Weighted Average						
1	0.1750	14450	0.1750	0.1900	0.1763	0.1508
2	0.1750	7333	0.1750	0.1933		
3	0.1650	12515	0.1650	0.1933	0.0304	0.0351
4*	0.1500	3520	0.1500	0.1667		
5	0.1750	13151	0.1750	0.1933		
6*	0.1450	10709	0.1450	0.1717		
7	0.1950	11960	0.1950	0.2067		
8*	0.1100	2744	0.1100	0.0983		
9	0.1900	12546	0.1900	0.1833		
10*	0.1400	12376	0.1400	0.1667		

Table 7: Six Hidden Node Output (continued)

				Val Set	Ave Error	Select
				Error	Rate or	Ave Er Rat
Cycle	Statistic	Epoch	Statistic	Rate	Std Dev	or St Dev
Moving Average						
1	0.2220	13846	0.2220	0.1917	0.1778	0.1458
2	0.2280	9527	0.2280	0.1950		
3	0.2200	12501	0.2200	0.1933	0.0361	0.0339
4*	0.1920	9268	0.1920	0.1633		
5	0.2140	5325	0.2140	0.1883		
6*	0.1840	1756	0.1840	0.1633		
7	0.2520	1503	0.2520	0.2333		
8*	0.1100	2791	0.1100	0.0950		
9	0.2220	12486	0.2220	0.1933		
10*	0.1900	9089	0.1900	0.1617		
Total Absolute Error						
1	27.0278	13998	27.0278	0.1917	0.1743	0.1479
2	27.2391	11172	27.2391	0.1933		
3	27.0533	12920	27.0533	0.1883	0.0298	0.0320
4*	21.0477	9262	21.0477	0.1633		
5	26.8933	14383	26.8933	0.1950		
6*	21.5338	6861	21.5338	0.1633		
7	28.7017	4507	28.7017	0.2033		
8*	15.9658	4120	15.9658	0.1000		
9	25.1477	12547	25.1477	0.1800		
10*	21.2537	10817	21.2537	0.1650		

Table 8: Eight Hidden Node Output

Cycle	Statistic	Epoch	Statistic	Val Set Error Rate	Ave Error Rate or Std Dev	Select Ave Er Rat or St Dev
Training Set						
1	0.0700	10623	0.0700	0.1367	0.1655	0.1267
2	0.1200	1560	0.1200	0.2117		
3*	0.0400	8544	0.0400	0.1050	0.0464	0.0390
4	0.0800	7283	0.0800	0.1683		
5	0.0700	7996	0.0700	0.1317		
6	0.1000	13383	0.1000	0.2283		
7*	0.0400	5114	0.0400	0.1033		
8	0.1100	12596	0.1100	0.2267		
9	0.0800	4876	0.0800	0.1717		
10*	0.0500	13401	0.0500	0.1717		
Testing Set						
1	0.1800	5600	0.1800	0.1267	0.1592	0.0967
2	0.2300	6459	0.2300	0.2117		
3*	0.1400	8028	0.1400	0.1050	0.0458	0.0118
4	0.1800	2530	0.1800	0.1650		
5	0.1900	2983	0.1900	0.1383		
6	0.2300	1635	0.2300	0.2150		
7*	0.0800	9827	0.0800	0.0883		
8	0.2300	6780	0.2300	0.2183		
9	0.1900	4199	0.1900	0.1667		
10	0.1800	1980	0.1800	0.1567		
Weighted Average						
1	0.1450	4977	0.1450	0.1333	0.1600	0.0992
2	0.1850	1560	0.1850	0.2117		
3*	0.0950	8544	0.0950	0.1050	0.0456	0.0082
4	0.1450	13710	0.1450	0.1583		
5	0.1400	7996	0.1400	0.1317		
6	0.1800	8609	0.1800	0.2217		
7*	0.0800	12393	0.0800	0.0933		
8	0.1900	2252	0.1900	0.2167		
9	0.1400	14239	0.1400	0.1617		
10	0.1400	11736	0.1400	0.1667		

Table 8: Eight Hidden Node Output (continued)

Cycle	Statistic	Epoch	Statistic	Val Set Error Rate	Ave Error Rate or Std Dev	Select Ave Er Rat or St Dev
Moving Average						
1	0.1920	10122	0.1920	0.1333	0.1598	0.1000
2	0.2400	8494	0.2400	0.2250		
3*	0.1400	8416	0.1400	0.1067	0.0476	0.0094
4	0.1880	14258	0.1880	0.1533		
5	0.1960	5704	0.1960	0.1300		
6	0.2400	8828	0.2400	0.2250		
7*	0.0900	2142	0.0900	0.0933		
8	0.2360	7113	0.2360	0.2150		
9	0.1900	6265	0.1900	0.1600		
10	0.1860	1984	0.1860	0.1567		
Total Absolute Error						
1	20.1558	7109	20.1558	0.1267	0.1592	0.1000
2	25.3412	10949	25.3412	0.1967		
3*	15.3280	13261	15.3280	0.1050	0.0465	0.0071
4	20.6984	14981	20.6984	0.1517		
5	20.7916	7965	20.7916	0.1283		
6	27.2794	14380	27.2794	0.2200		
7*	12.9566	12416	12.9566	0.0950		
8	27.7513	13800	27.7513	0.2300		
9	21.2363	12660	21.2363	0.1683		
10	22.2873	14104	22.2873	0.1700		

Table 9: Ten Hidden Node Output

Cycle	Statistic	Epoch	Statistic	Val Set Error Rate	Ave Error Rate or Std Dev	Select Ave Er Rat or St Dev
Training Set						
1*	0.0400	14314	0.0400	0.0883	0.1477	0.0894
2	0.1100	14956	0.1100	0.1750		
3*	0.0400	14646	0.0400	0.0817	0.0457	0.0386
4	0.0900	13296	0.0900	0.1733		
5	0.0700	10135	0.0700	0.1550		
6	0.1100	12183	0.1100	0.2100		
7	0.0700	7521	0.0700	0.1517		
8*	0.0400	9619	0.0400	0.0983		
9	0.0600	5310	0.0600	0.1400		
10	0.0800	6088	0.0800	0.2033		
Testing Set						
1*	0.0900	10849	0.0900	0.0800	0.1437	0.0861
2	0.2200	11701	0.2200	0.1717		
3*	0.0800	6059	0.0800	0.0933	0.0446	0.0067
4	0.2000	4383	0.2000	0.1850		
5	0.1800	5358	0.1800	0.1283		
6	0.2200	2868	0.2200	0.2017		
7	0.1900	2844	0.1900	0.1533		
8*	0.0900	12581	0.0900	0.0850		
9	0.1700	1779	0.1700	0.1550		
10	0.2200	3164	0.2200	0.1833		
Weighted Average						
1*	0.0800	12577	0.0800	0.0950	0.1448	0.0917
2	0.1750	14956	0.1750	0.1750		
3*	0.0850	6272	0.0850	0.0850	0.0424	0.0058
4	0.1600	5758	0.1600	0.1833		
5	0.1400	6329	0.1400	0.1350		
6	0.1800	14428	0.1800	0.2050		
7	0.1400	10518	0.1400	0.1517		
8*	0.0850	14394	0.0850	0.0950		
9	0.1300	5310	0.1300	0.1400		
10	0.1800	3392	0.1800	0.1833		

Table 9: Ten Hidden Node Output (continued)

				Val Set	Ave Error	Select
				Error	Rate or	Ave Er Rat
Cycle	Statistic	Epoch	Statistic	Rate	Std Dev	or St Dev
Moving Average						
1*	0.1000	13057	0.1000	0.0900	0.1455	0.0900
2	0.2340	12268	0.2340	0.1783		
3*	0.0920	6068	0.0920	0.0867	0.0441	0.0033
4	0.2100	4386	0.2100	0.1900		
5	0.1960	6333	0.1960	0.1383		
6	0.2320	2869	0.2320	0.2067		
7	0.1960	14940	0.1960	0.1383		
8*	0.0960	12585	0.0960	0.0933		
9	0.1700	1978	0.1700	0.1517		
10	0.2200	3337	0.2200	0.1817		
Total Absolute Error						
1*	13.4608	13345	13.4608	0.0883	0.1450	0.0883
2	27.3778	12622	27.3778	0.1767		
3*	13.0426	9083	13.0426	0.0883	0.0477	0.0000
4	24.3949	14578	24.3949	0.1767		
5	21.0033	7629	21.0033	0.1317		
6	27.1659	14722	27.1659	0.2017		
7	21.3834	14936	21.3834	0.1350		
8*	13.3277	14892	13.3277	0.0883		
9	20.2756	12328	20.2756	0.1450		
10	28.3567	12965	28.3567	0.2183		

Table 10: Twelve Hidden Node Output

				Val Set	Ave Error	Select
				Error	Rate or	Ave Er Rat
Cycle	Statistic	Epoch	Statistic	Rate	Std Dev	or St Dev
Training Set						
1*	0.040	5120	0.040	0.0983	0.1517	0.1253
2*	0.050	4331	0.050	0.1600		
3*	0.060	8676	0.060	0.1417	0.0438	0.0322
4	0.080	10609	0.080	0.2150		
5*	0.050	11658	0.050	0.1600		
6	0.110	10718	0.110	0.1850		
7	0.090	4073	0.090	0.2033		
8*	0.050	13766	0.050	0.1000		
9*	0.040	3997	0.040	0.0917		
10	0.070	4663	0.070	0.1617		
Testing Set						
1*	0.080	2344	0.080	0.0983	0.1433	0.0933
2	0.190	1386	0.190	0.1767		
3	0.150	2411	0.150	0.1383	0.0404	0.0101
4	0.200	2071	0.200	0.1817		
5	0.200	3565	0.200	0.1517		
6	0.230	5501	0.230	0.1850		
7	0.220	2501	0.220	0.1917		
8*	0.090	14588	0.090	0.0817		
9*	0.060	3033	0.060	0.1000		
10	0.180	8372	0.180	0.1283		
Weighted Average						
1*	0.080	5120	0.080	0.0983	0.1437	0.0944
2	0.135	1953	0.135	0.1517		
3	0.130	2222	0.130	0.1500	0.0379	0.0035
4	0.170	2370	0.170	0.1833		
5	0.145	3584	0.145	0.1617		
6	0.180	12804	0.180	0.1717		
7	0.190	2536	0.190	0.1967		
8*	0.090	14733	0.090	0.0933		
9*	0.070	3997	0.070	0.0917		
10	0.140	9873	0.140	0.1383		

Table 10: Twelve Hidden Node Output (continued)

Cycle	Statistic	Epoch	Statistic	Val Set Error Rate	Ave Error Rate or Std Dev	Select Ave Er Rat or St Dev
Moving Average						
1*	0.086	3180	0.086	0.0983	0.1433	0.1025
2	0.190	1702	0.190	0.1600		
3	0.164	2484	0.164	0.1350	0.0371	0.0059
4	0.200	2371	0.200	0.1767		
5	0.204	3568	0.204	0.1567		
6	0.246	10999	0.246	0.1767		
7	0.230	2509	0.230	0.1983		
8	0.108	14592	0.108	0.0867		
9*	0.074	2429	0.074	0.1067		
10	0.194	13562	0.194	0.1383		
Total Absolute Error						
1*	11.4825	13797	11.4825	0.0867	0.1385	0.0817
2	22.4609	4793	22.4609	0.1517		
3	17.8558	7120	17.8558	0.1367	0.0442	0.0044
4	27.6511	2908	27.6511	0.1883		
5	24.0679	3856	24.0679	0.1533		
6	27.8478	12823	27.8478	0.1900		
7	28.2915	5535	28.2915	0.1867		
8*	14.6244	14642	14.6244	0.0783		
9*	12.1035	6003	12.1035	0.0800		
10	20.1519	12839	20.1519	0.1333		

Table 11: Fourteen Hidden Node Output

Cycle	Statistic	Epoch	Statistic	Val Set Error Rate	Ave Error Rate or Std Dev	Select Ave Er Rat or St Dev
Training Set						
1	0.070	14252	0.070	0.2133	0.1593	0.1504
2*	0.060	12640	0.060	0.1717		
3*	0.050	10388	0.050	0.1583	0.0269	0.0195
4*	0.060	12171	0.060	0.1300		
5	0.090	9264	0.090	0.1767		
6*	0.060	9614	0.060	0.1550		
7*	0.060	9878	0.060	0.1450		
8*	0.060	3681	0.060	0.1683		
9*	0.050	6284	0.050	0.1150		
10*	0.060	7325	0.060	0.1600		
Testing Set						
1	0.230	2775	0.230	0.2050	0.1487	0.1333
2	0.200	3142	0.200	0.1600		
3*	0.160	2266	0.160	0.1400	0.0312	0.0284
4*	0.160	14069	0.160	0.1167		
5*	0.170	7255	0.170	0.1633		
6	0.200	9747	0.200	0.1500		
7	0.180	4944	0.180	0.1300		
8	0.180	1579	0.180	0.1750		
9*	0.110	6161	0.110	0.0933		
10*	0.160	2726	0.160	0.1533		
Weighted Average						
1	0.180	13910	0.180	0.1900	0.1518	0.1278
2	0.145	3131	0.145	0.1667		
3*	0.125	5354	0.125	0.1533	0.0247	0.0231
4*	0.125	13771	0.125	0.1217		
5	0.140	10468	0.140	0.1600		
6	0.140	14571	0.140	0.1483		
7	0.145	4971	0.145	0.1333		
8	0.150	1720	0.150	0.1733		
9*	0.090	12326	0.090	0.1083		
10	0.145	2839	0.145	0.1633		

Table 11: Fourteen Hidden Node Output (continued)

Cycle	Statistic	Epoch	Statistic	Val Set Error Rate	Ave Error Rate or Std Dev	Select Ave Er Rat or St Dev
Moving Average						
1	0.244	2964	0.244	0.2067	0.1517	0.1194
2	0.208	3145	0.208	0.1633		
3*	0.170	2270	0.170	0.1417	0.0309	0.0210
4*	0.166	14989	0.166	0.1167		
5	0.182	7282	0.182	0.1667		
6	0.214	9083	0.214	0.1500		
7	0.188	4947	0.188	0.1317		
8	0.188	1581	0.188	0.1783		
9*	0.116	8997	0.116	0.1000		
10	0.186	2730	0.186	0.1617		
Total Absolute Error						
1	27.8979	13187	27.8979	0.2017	0.1493	0.1183
2	24.0182	12284	24.0182	0.1650		
3*	17.9408	4484	17.9408	0.1350	0.0288	0.0176
4*	18.7044	14986	18.7044	0.1200		
5	20.4197	7674	20.4197	0.1633		
6	23.0643	6470	23.0643	0.1650		
7	20.2532	13232	20.2532	0.1300		
8	22.2750	6451	22.2750	0.1617		
9*	12.2425	9226	12.2425	0.1000		
10	22.1440	2792	22.1440	0.1517		

Table 12: Sixteen Hidden Node Output

Cycle	Statistic	Epoch	Statistic	Val Set Error Rate	Ave Error Rate or Std Dev	Select Ave Er Rat or St Dev
Training Set						
1*	0.0600	9797	0.0600	0.1567	0.1415	0.1381
2*	0.0500	10465	0.0500	0.1100		
3*	0.0600	6068	0.0600	0.1617	0.0291	0.0288
4*	0.0500	7361	0.0500	0.1733		
5*	0.0500	12999	0.0500	0.1500		
6*	0.0600	10537	0.0600	0.1600		
7*	0.0500	4389	0.0500	0.1017		
8*	0.0400	8959	0.0400	0.1333		
9	0.0800	13681	0.0800	0.1717		
10*	0.0500	7032	0.0500	0.0967		
Testing Set						
1	0.1700	3268	0.1700	0.1433	0.1293	0.1133
2*	0.1300	12350	0.1300	0.1050		
3	0.1900	14776	0.1900	0.1450	0.0314	0.0145
4	0.2100	3584	0.2100	0.1650		
5*	0.1500	9371	0.1500	0.1350		
6	0.1700	2910	0.1700	0.1400		
7*	0.1200	6386	0.1200	0.1050		
8*	0.1400	10793	0.1400	0.1083		
9	0.2100	1620	0.2100	0.1750		
10	0.0700	14970	0.0700	0.0717		
Weighted Average						
1	0.1300	3354	0.1300	0.1383	0.1343	0.1120
2*	0.1050	11985	0.1050	0.1133		
3	0.1400	14520	0.1400	0.1467	0.0299	0.0239
4	0.1450	7361	0.1450	0.1733		
5*	0.1150	12999	0.1150	0.1500		
6	0.1350	2173	0.1350	0.1617		
7*	0.1000	4389	0.1000	0.1017		
8*	0.1150	10870	0.1150	0.1100		
9	0.1600	8522	0.1600	0.1633		
10*	0.0750	14701	0.0750	0.0850		

Table 12: Sixteen Hidden Node Output (continued)

				Val Set	Ave Error	Select
				Error	Rate or	Ave Er Rat
Cycle	Statistic	Epoch	Statistic	Rate	Std Dev	or St Dev
Moving Average						
1	0.1700	3334	0.1700	0.1383	0.1298	0.0971
2*	0.1360	13962	0.1360	0.0950		
3	0.2040	14725	0.2040	0.1417	0.0305	0.0103
4	0.2100	3926	0.2100	0.1700		
5	0.1640	10265	0.1640	0.1400		
6	0.1800	2133	0.1800	0.1600		
7*	0.1280	6386	0.1280	0.1050		
8*	0.1480	10797	0.1480	0.1050		
9	0.2120	1982	0.2120	0.1600		
10*	0.0880	2743	0.0880	0.0833		
Total Absolute Error						
1	19.3845	3480	19.3845	0.1400	0.1325	0.1100
2*	14.8096	14372	14.8096	0.1017		
3	22.3533	14609	22.3533	0.1500	0.0292	0.0218
4	23.0003	8420	23.0003	0.1683		
5*	16.9740	12140	16.9740	0.1367		
6	21.8437	2557	21.8437	0.1467		
7*	15.0910	10962	15.0910	0.1033		
8*	17.0449	3778	17.0449	0.1267		
9	23.8317	1978	23.8317	0.1700		
10*	12.9401	14971	12.9401	0.0817		

Table 13: Eighteen Hidden Node Output

				Val Set	Ave Error	Select
				Error	Rate or	Ave Er Rat
Cycle	Statistic	Epoch	Statistic	Rate	Std Dev	or St Dev
Training Set						
1*	0.0600	3510	0.0600	0.1500	0.1293	0.1254
2*	0.0500	6250	0.0500	0.1233		
3	0.0800	9273	0.0800	0.1650	0.0186	0.0146
4*	0.0500	8789	0.0500	0.1250		
5*	0.0500	13017	0.0500	0.1200		
6*	0.0500	7727	0.0500	0.1367		
7*	0.0400	8655	0.0400	0.1067		
8*	0.0500	4960	0.0500	0.1133		
9*	0.0500	4288	0.0500	0.1417		
10*	0.0400	5383	0.0400	0.1117		
Testing Set						
1	0.1600	2418	0.1600	0.1483	0.1172	0.0950
2*	0.0900	13602	0.0900	0.0833		
3	0.1700	10942	0.1700	0.1500	0.0281	0.0180
4*	0.1000	11700	0.1000	0.0850		
5	0.1400	11198	0.1400	0.1067		
6	0.1600	3722	0.1600	0.1300		
7*	0.1000	10932	0.1000	0.0900		
8*	0.1000	2206	0.1000	0.1217		
9	0.1600	2426	0.1600	0.1567		
10	0.1200	11988	0.1200	0.1000		
Weighted Average						
1	0.1250	5392	0.1250	0.1333	0.1147	0.0977
2*	0.0900	12769	0.0900	0.0933		
3	0.1350	11610	0.1350	0.1550	0.0231	0.0163
4*	0.0850	14917	0.0850	0.0717		
5	0.1100	12743	0.1100	0.1150		
6	0.1250	3737	0.1250	0.1283		
7*	0.0850	9390	0.0850	0.1017		
8*	0.0950	2548	0.0950	0.1100		
9	0.1200	13327	0.1200	0.1267		
10*	0.1000	5383	0.1000	0.1117		

Table 13: Eighteen Hidden Node Output (continued)

Cycle	Statistic	Epoch	Statistic	Val Set Error Rate	Ave Error Rate or Std Dev	Select Ave Er Rat or St Dev
Moving Average						
1	0.1620	5394	0.1620	0.1300	0.1143	0.0937
2*	0.1100	13605	0.1100	0.0883		
3	0.1780	10946	0.1780	0.1533	0.0252	0.0122
4*	0.1060	14123	0.1060	0.0750		
5	0.1480	13188	0.1480	0.1183		
6	0.1600	3741	0.1600	0.1267		
7*	0.1120	14267	0.1120	0.0983		
8*	0.1060	2565	0.1060	0.1050		
9	0.1600	2757	0.1600	0.1467		
10*	0.1200	14146	0.1200	0.1017		
Total Absolute Error						
1	16.9836	5399	16.9836	0.1350	0.1123	0.0900
2*	12.0965	13602	12.0965	0.0833		
3	19.3853	10958	19.3853	0.1567	0.0271	0.0115
4*	12.1219	13297	12.1219	0.0733		
5	17.2198	11205	17.2198	0.1133		
6	19.1026	3743	19.1026	0.1250		
7*	12.5790	14263	12.5790	0.0933		
8*	13.9738	13069	13.9738	0.1000		
9	17.9447	3920	17.9447	0.1433		
10*	13.6861	11988	13.6861	0.1000		

Table 14: Twenty Hidden Node Output

Cycle	Statistic	Epoch	Statistic	Val Set Error Rate	Ave Error Rate or Std Dev	Select Ave Er Rat or St Dev
Training Set						
1	0.0600	3113	0.0600	0.1533	0.1298	0.1117
2*	0.0400	14840	0.0400	0.1100		
3*	0.0400	4677	0.0400	0.1167	0.0233	0.0234
4	0.0600	3629	0.0600	0.1417		
5	0.0500	7143	0.0500	0.1633		
6	0.0500	3610	0.0500	0.1267		
7	0.0500	6887	0.0500	0.1267		
8*	0.0300	7928	0.0300	0.1383		
9	0.0500	8589	0.0500	0.1400		
10*	0.0200	9921	0.0200	0.0817		
Testing Set						
1	0.1600	1803	0.1600	0.1600	0.1270	0.0850
2*	0.1100	13293	0.1100	0.0983		
3*	0.1100	8932	0.1100	0.0683	0.0350	0.0153
4	0.1400	2264	0.1400	0.1433		
5	0.2000	2316	0.2000	0.1867		
6	0.1300	2077	0.1300	0.1317		
7	0.1400	10195	0.1400	0.1183		
8	0.1500	7693	0.1500	0.1333		
9	0.1700	5058	0.1700	0.1417		
10*	0.0700	2638	0.0700	0.0883		
Weighted Average						
1	0.1300	3113	0.1300	0.1533	0.1233	0.0906
2*	0.0900	12944	0.0900	0.1083		
3*	0.0850	8922	0.0850	0.0817	0.0284	0.0154
4	0.1250	2264	0.1250	0.1433		
5	0.1400	7143	0.1400	0.1633		
6	0.1100	2229	0.1100	0.1250		
7	0.1050	13571	0.1050	0.1050		
8	0.1100	7928	0.1100	0.1383		
9	0.1250	7411	0.1250	0.1333		
10*	0.0600	9921	0.0600	0.0817		

Table 14: Twenty Hidden Node Output (continued)

Cycle	Statistic	Epoch	Statistic	Val Set Error Rate	Ave Error Rate or Std Dev	Select Ave Er Rat or St Dev
Moving Average						
1	0.1680	2222	0.1680	0.1500	0.1255	0.0872
2*	0.1180	13297	0.1180	0.1083		
3*	0.1120	8936	0.1120	0.0733	0.0336	0.0186
4	0.1560	2264	0.1560	0.1433		
5	0.2000	2370	0.2000	0.1867		
6	0.1380	2077	0.1380	0.1317		
7	0.1420	12431	0.1420	0.1117		
8	0.1580	7693	0.1580	0.1333		
9	0.1780	5060	0.1780	0.1367		
10*	0.0840	9292	0.0840	0.0800		
Total Absolute Error						
1	19.1617	3763	19.1617	0.1350	0.1197	0.0833
2*	15.5814	13294	15.5814	0.1067		
3*	12.2777	8934	12.2777	0.0700	0.0306	0.0203
4	17.8040	4848	17.8040	0.1383		
5	23.5986	9261	23.5986	0.1650		
6	16.6230	2711	16.6230	0.1183		
7	15.0833	13539	15.0833	0.1100		
8	18.5888	9806	18.5888	0.1467		
9	18.4043	6421	18.4043	0.1333		
10*	11.6428	14751	11.6428	0.0733		

Vita

Captain Bruce Kostal was born on 22 September 1959 in Beatrice, Nebraska. He was graduated from the University of Nebraska in 1982 with a Bachelor of Science degrees in Business Administration (Accounting). After graduation, he was assigned to Whiteman AFB, Missouri as a Missile Launch Officer. There, he also received a Master of Business Administration (Personnel Administration) degree and a Certified Public Accounting Certificate (State of Nebraska). In 1987, he was transferred to Vandenberg AFB, California, as a member of the Top Hand program, which is responsible for operational missile test launches. In August 1992, he entered the Strategic and Tactical Sciences Program, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio.

Captain Kostal is married to the former Donna Jurgens of Adams, Nebraska. They have two daughters, Kara and Whitney.

Permanent address: Route One, Box 230
Wymore, Nebraska 68466

Bibliography

1. Belue, Capt Lisa M. *An Investigation of Multilayer Perceptrons for Classification.* MS thesis, AFIT/GOR/ENS/92M-06. School of Engineering, Air Force Institute of Technology(AU), Wright-Patterson AFB OH, March 1992.

2. Choie, YoungJu and others. "On Benchmarks for Learning Algorithms," *1991 IEEE International Joint Conference on Neural Networks, 1* 723-728. New York: IEEE Press, 1989.

3. Guyon, I. "Neural Networks and Applications Tutorial," *Physics Reports, 207* (3-5): 215-259 (September 1991).

4. Hart, Anna. "Using Neural Networks for Classification Tasks–Some Experiments on Datasets and Practical Advice," *OR: Journal of the Operational Research Society, 43* (3): 215-226 (March 1992).

5. Choie, YoungJu and others. "On Benchmarks for Learning Algorithms," *1991 IEEE International Joint Conference on Neural Networks, 1* 723-728. New York: IEEE Press, 1989.

6. Hergert, F. and others. "A Comparison of Weight Elimination methods for Reducing Complexity in Neural Networks," *International Joint Conference on Neural Networks, 3* 980-987. New York: IEEE Press, 1992.

7. Müller, B. and J. Reinhardt. *Neural Networks: An Introduction.* Berlin: Springer-Verlag, 1990.

8. Nelson, Marilyn McCord and W. T. Illingworth. *A Practical Guide to Neural Nets.* Reading, MA: Addison-Wesley Publishing Company, Inc., 1991.

9. Pan, H.-L. and Y.-C. Chen. "Liver Tissues Classification by Artificial Neural Networks," *Pattern Recognition Letters, 13* (5): 355-368 (May 1992).

10. Ramamoorthy, C. V. and S. Shekhar. "A Stochastic Learning Algorithm for Generalization Problems," *TENCON '89 Fourth IEEE Region 10 International Conference. 'Information Technologies for the 90's'* 136-141. New York: IEEE Press, 1989.

11. Rogers, Steven K., Matthew Kabrisky, Dennis W. Ruck, and Gregory L. Tarr. *An Introduction to Biological and Artificial Neural Networks.* Unpublished Report. Air Force Institute of Technology, Wright-Patterson Air Force Base, OH, 1990.

12. Tou, J. T. and R. C. Gonzalez. *Pattern Recognition Principles.* Reading, MA: Addison-Wesley Publishing Company, 1974.

13. Weiss, Sholom M. and Casimir A. Kulikowski. *Computer Systems That Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems.* San Mateo, CA: Morgan Kaufmann Publishers, Inc., 1991.

14. Wiggins, Vince L. and others. *Neural Networks: A Primer.* Contract F41689-88-D-0251. Human Resources Directorate, Manpower and Personnel Research Division, Brooks Air Force Base, TX, May 1991.

REPORT DOCUMENTATION PAGE

OMB No. 0704-0188

[illegible] data sources [illegible] the data needed and completing and reviewing the [illegible] Send comments regarding this burden estimate or any other aspect of this [illegible] suggestions for reducing this burden to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson [illegible] Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (*Leave blank*)

2. REPORT DATE
March 1994

3. REPORT TYPE AND DATES COVERED
Master's Thesis

4. TITLE AND SUBTITLE
AN ANALYSIS OF STOPPING CRITERIA
IN ARTIFICIAL NEURAL NETWORKS

5. FUNDING NUMBERS

6. AUTHOR(S)
Captain Bruce E. Kostal, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology, WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GST/ENG/94M-07

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

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 goal of this study was to decide when to terminate training of an artificial neural network (ANN). In pursuit of this goal, several characteristics of the ANN were monitored throughout ANN training: classification error rate (of the training set, testing set, or a weighted average of the two); moving average classification error rate; measurements of the difference between ANN output and desired output (error sum of squares, total absolute error, or largest absolute error); or ANN weight changes (absolute weight change, squared weight change, or relative weight change). Throughout this research, the learning rate was held constant at 0.35. The momentum was not used because the primary interest of this study was evaluating when to terminate training as opposed to the speed of reaching a decision. The ANN structure was held constant with two input features, two output classes, and one hidden layer. Finally, the practice of training after processing each exemplar was followed. Results indicated three conclusions. First, multiple runs are required for ANN analysis because ANNs are not guaranteed to converge to the same solution. Second, the classification error rate, a moving average of the classification error rate, and the total absolute error (all computed on the testing set) provided a similar final classification. Third, once the stopping criteria functions cease to decrease, select a set of ANN final weights at random. On average, they yielded as low as or lower classification error rate and variance of the classification error rate than other possible choices.

14. SUBJECT TERMS
Artificial neural networks, stopping criteria

15. NUMBER OF PAGES
159

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT