A TEST OF NOISE FILTERING CAPABILITIES OF BACKPROPAGATION NEURAL NETWORKS

By

RANGANATHAN RAMKUMAR

Bachelor of Engineering

P. S. G. College of Technology

Coimbatore, India

1986

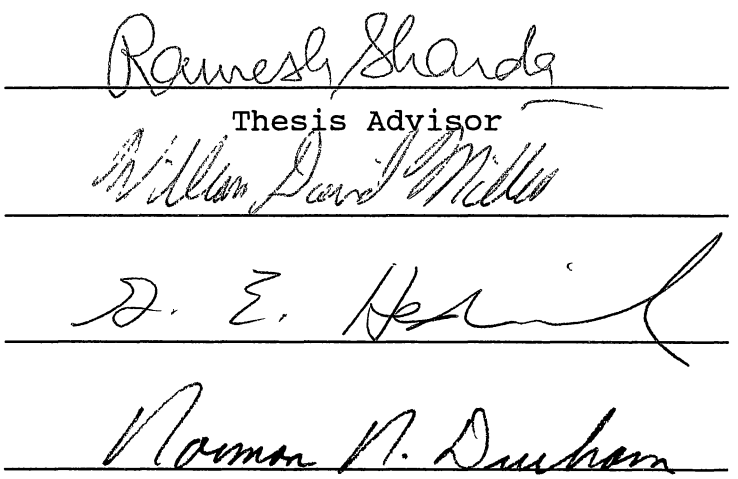
Submitted to the faculty of the
Graduate College of the
Oklahoma State University
In partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
May, 1990

Thesis
1990
R173t
cop.2

A TEST OF NOISE FILTERING CAPABILITIES

OF BACKPROPAGATION NEURAL NETWORKS

Thesis Approved:



Thesis Advisor

Dean of the Graduate College

1366857

ACKNOWLEDGMENTS

This writer wishes to express his grateful appreciation to Dr. Ramesh Sharda for his interest, direction, wisdom, and counsel during the conduct of this study and writing of this thesis. The writer also wishes to thank Dr. David W. Miller and Dr. George E. Hedrick for aiding and helping as members of the thesis committee.

Additionally, the writer wishes to thank Mr. John Hart and Mr. Frank Boyer of Frontier Engineering Incorporated for their suggestions and advice during the course of this research. Finally, the writer wishes to express his gratitude to Dr. Robert C. Dauffenbach, Director of Business and Economic Research at Oklahoma State University for his support by employing the author as a graduate research assistant.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I

INTRODUCTION

General Statement of the Problem

Neural Networks have long been studied in an attempt to explain and to exploit computational abilities of massively parallel systems. This effort has stemmed, in part, from the assumption that biological neural systems operate in a parallel manner and from the observation that classical computer architectures may be insufficient for the performance of some tasks. Further, these very tasks, pattern recognition for one, are exactly those at which complex biological systems excel. As physically imposed constraints increasingly limit the performance of standard serial architectures, different neural models have been proposed in an attempt to duplicate the behavior of real biological neural systems. A side benefit of this intense interest in neural networks has been the emergence of structures with ability to do complex processing in a massively parallel way.

Filtering is concerned with the isolation or retrieval

of signals from noise. Typical applications include:

* Reception and discrimination of radio signals
* Digital data transmission on telephone lines
* Beam-forming arrays
* Detection of radar signals
* Processing of pictures sent from spacecraft
* Analysis of electrocardiogram (EKG) and electroencephalogram (EEG) signals

All of these are ideal tasks for a neural network based approach. In this research we consider the isolation or detection of EKG signals from their noisy environment. The noise is assumed to be random and of different frequency than the underlying signal. If the signal being worked on has a noise component added to it, the contribution of the noise to the signal will be random with respect to the other features of the original signal [KLIMASAUSKAS, 1989]. The noise in EKG signals is caused by fluorescent lights. The use of neural networks is in the development of a limited number of "feature detectors" that can account for the maximum amount of a signal.

The purpose of this study is to implement backpropagation neural network techniques for adaptive filtering using artificial neural simulation software and to identify the underlying signals. By varying the elements in the hidden layers the performance of the model can be

changed. The neural network technique is expected to improve the signal to noise ratio and reproduction of the wave pattern over the conventional methods.

The general approach to use neural networks for such applications is: (1) to select an appropriate type of signal for training and testing the neural network filter; (in this study EKG signals are used) (2) to build a suitable neural network model by choosing an appropriate learning paradigm and determining the correct number of processing elements in each layer; (3) to train the network with appropriate momentum and learning rate; (4) to modify the number of units in the hidden layer in an effort to reconstruct the original signal retaining the highest level of details; (5) to test the model with noisy input after suitable formatting.

Approach to the Problem

The approach taken here is to find a way of encoding or compressing the input data and then re-expanding it. The compression process eliminates portions of the input data which represent small or non-recurring features. By selecting the number of "encoders" the amount of detail retained in the transformation can be varied. This form of data compression is called dimensionality reduction [KLIMASAUSKAS,1989]. To achieve this reduction it is

important that the following criteria be met [KLIMASAUSKAS, 1989]:

* There must be some form of relationship between input variables.
* An encoding system must be used that can represent the relationship.

If there is no definite relationship among the input variables, dimensionality reduction often results in a series of fixed outputs which represent the average value of a combination of inputs. In noise filtering a high degree of relationship between adjacent time samples exists. Dimensionality reduction usually works well in this situation.

As described earlier, the contribution of noise to a signal will be random when compared with the prominent features. If the noise component is of a relatively low frequency and accounts for a majority of the energy in the output signal (for example, 60 cycle hum) this process may detect noise. The novel characteristic of the neural network approach is that the weighted linear combination of inputs is transformed using a nonlinear function such as a sigmoid or sine function. These nonlinearities make the creation of multilevel systems possible and are responsible for several of the resulting output characteristics.

Why Neural Networks?

The answer to this question can be given by two basic reasons. First, neural networks are capable of retaining a greater level of detail than other techniques. Rather than simply ignoring small features, they allow them to pass if they are of a stationary recurring nature. Second, the amount of detail retained by the filtering process can be varied by changing the number of elements in the hidden layer of the neural model. From these points of view, neural network techniques provide additional flexibility and control when the user analyzes noisy data.

The approach, as mentioned earlier, is to develop a few elements (feature detectors) that encode a series of samples of an input signal. To reconstruct all or a portion of the input sample, the output of these encoders (processing elements in neural network sense) is used. This approach of using a few detectors ensures that some of the information (especially the noise) will be removed from the reconstructed signal. One of the interesting characteristics of neural networks is their ability to develop adaptive filtering techniques that can be tuned to preserve varying degrees of detail [KLIMASAUSKAS, 1989].

EKG Signals - A Case Study

The data for studying the neural network ability in noise filtering are EKG signals. The data have been obtained from a local firm. These signals are unfiltered and contain the actual EKG data with noise and other spurious signals. The backpropagation technique of the neural network model was implemented to filter the noise. Backpropagation is a very powerful adaptive technique for approximating relationships between several continuous-valued inputs and outputs. This analysis was done on Artificial Neural Networks Simulation Software (ANSIM).

Neural Network Simulation Tool

Many attributes were considered for selecting the neural network simulation tool: cost, size limit, functional capabilities, speed, ease of learning, interfaces to other software, portability, documentation, training, paradigms supported, company support and satisfaction. Not all tools are good for all kinds of functions. Each is best suited for a particular kind of application. Science Applications Integrated Company's artificial neural simulation software, ANSIM appears the best for the following reasons. It supports the backpropagation technique and gives freedom in

interfacing external data files. It makes it easy to change the network design, provides interfaces to assist in setting up the problem, and has tools for investigating what is happening within the network itself. In essence this software provides the tool required to solve the problem. The main drawback of this software is its inability to accept interactive input from the keyboard. The input must always be presented from a file.

Objectives of the Study

The objectives of the study are to 1) design, implement, and test a neural network based signal filter for noise filtering of EKG signals; 2) study the effect of the number of units in the hidden layer on the reconstructed output signal; 3) observe the effect of the learning rate on the training of the neural networks; 4) study the effect of the number of training cycles on the output unit error.

In the area of signal processing, neural networks have been tried with some success, although there are no practical systems in the field yet. The adaptive combiner discussed later points out the usefulness of adaptive systems for certain signal processing problems. Neural networks can provide an adaptive, nonlinear capability that can be useful in some signal processing applications. Theoretical approaches to nonlinear filtering are very

difficult, and the adaptive system approach may again prove, as in the adaptive combiner, to be a very practical way to solve some problems. Therefore, even though no such signal processing applications are known at present, the capabilities of neural networks in providing adaptive, nonlinear signal processing will provide solution to some class of problems in the future. The hardware implementation of a neural network solution should also benefit in solving complex problems.

Scope and Limitations of the Study

The scope of this system is limited to identifying the signal embedded in random disturbances and was without real time data. The neural network filter attempts to reconstruct the original signal while retaining higher levels of detail based on the user's preferences.

The system assumed (1) the noise present in the input is random; (2) the original signal is of different frequency than the noise frequency; (3) the output of the Finite Impulse Filter (FIR), used as target vector during the training of the network is a pure signal; (4) the signal pattern is periodic; it keeps on repeating with constant time intervals; and (5) the number of samples constituting a wave pattern is constant. The entire model is simulated with the input supplied from a file.

CHAPTER II

RELATED STUDIES

Neural Networks

The interest in Artificial Neural Systems (ANS) has grown in tremendous proportions recently. These systems are also called Neural Networks, Connectionist Systems and Neurocomputers, and are spotlighted in Japan's most recent announcement of their sixth generation project [CAUDILL, 1987]. A neural network is a computing system made up of several simple, highly interconnected processing elements, which processes information by its dynamic state response to external inputs [HECHT-NIELSEN, 1986]. In a serial computing system, the execution is essentially sequential; all operations are executed in deterministic sequence. On the other hand the key to neural networks as an alternative to Von Neumann computing is their ability to "learn" and "adapt" themselves to survive in a constantly changing environment. Neural networks take in and analyze or process millions of possibilities. In neural networks the final output is not confined to one single location but is spread throughout the system.

Since their reemergence in the middle eighties, the field of neural networks has been developing rapidly. This is due to the advancement in hardware and software. Neural net research has a very strong foundation in mathematics. New concepts in the mathematics of neural models accompanied these developments [TANK and HOPFIELD, 1987]. A very good overview of the field and a description of some of the most studied models is provided in [LIPPMAN, 1987], while [RUMELHART AND McCLELLAND, 1985] provide a good theoretical background on the subject. Specific neural network models are studied in [HOPFIELD 1987], [ABILITIS, 1982], [CAUDILL, 1987-1989], [KOSKO, 1987], [WERBOS, 1989] and [HECHT-NIELSEN, 1989].

A neural network is modeled on the structure of the brain, but the neural networks used by researchers today are only loosely based upon biology. This is due to our inability to fully understand how a brain works. A neural network model consists of a collection of processing elements, called neurons, after their counterparts in the brain [HECHT-NIELSEN, 1987]. Each neuron can have several input signals coming to it either from other elements or from the external world, but there is only one output signal emerging from a single processing element. This output signal may fan out along many pathways to become one of the input signals for other elements (Figure 1). The processing

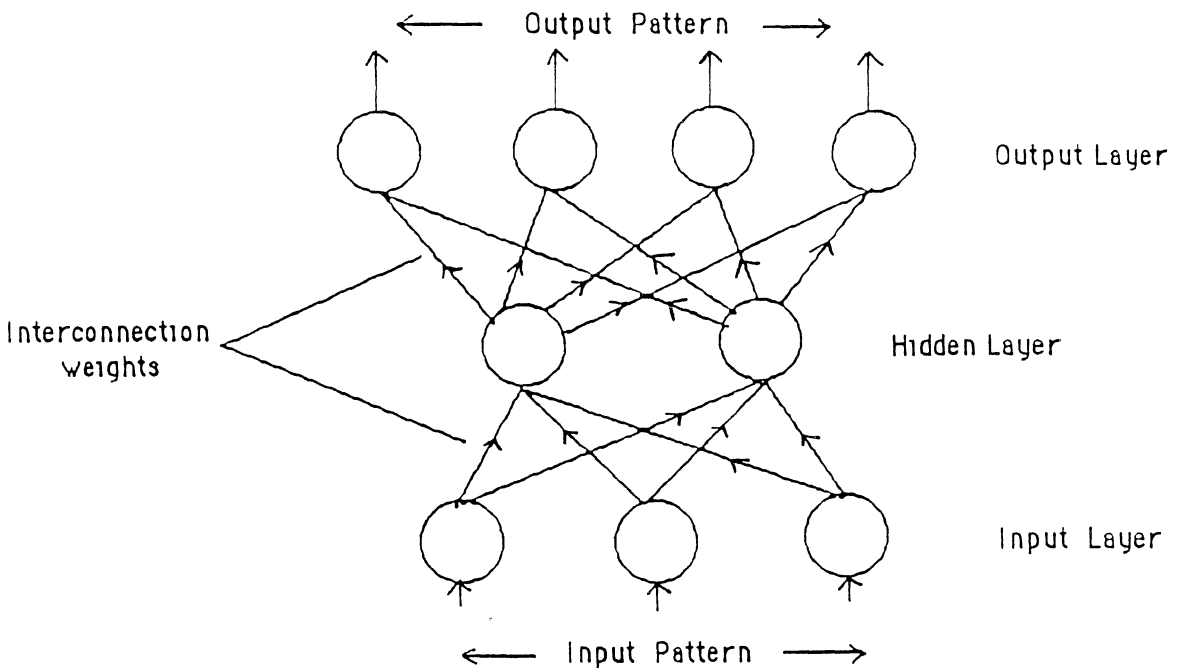


Figure 1. An Artificial Neural Network Model

that each element does is determined by its transfer function (Figure 2). Often a neural network model is divided into layers - groups of processing elements all having similar transfer functions [DARPA STUDY, 1988]. Every connection which enters a processing element has an adaptive coefficient, called a weight, assigned to it. The transfer function adds up all the weighted inputs to determine the value of its output. Thus weights determine the strengths of the connections from neighboring elements [CAUDILL, 1988].

There are several models of neural networks for fixed patterns. One summary of these models is in [LIPPMAN, 1987], based on input, either continuous valued or binary, and method of training, supervised or unsupervised (Figure 3). Of all the models, the backpropagation method of RUMELHART and McCLELLAND [1986] is perhaps the common one. It is a powerful model for signal processing applications.

Neural networks are being used in various fields. Widrow [1975] surveyed various potential applications of neural networks. Neural network models have great potential in areas such as speech recognition, image recognition, and pattern recognition, and other areas where many hypotheses are pursued in parallel and high computation rates are required and current systems are far from equaling human performance. They are used in robot control [WERBOS, 89], automotive diagnostics [MAKO, et.al., 1989], financial

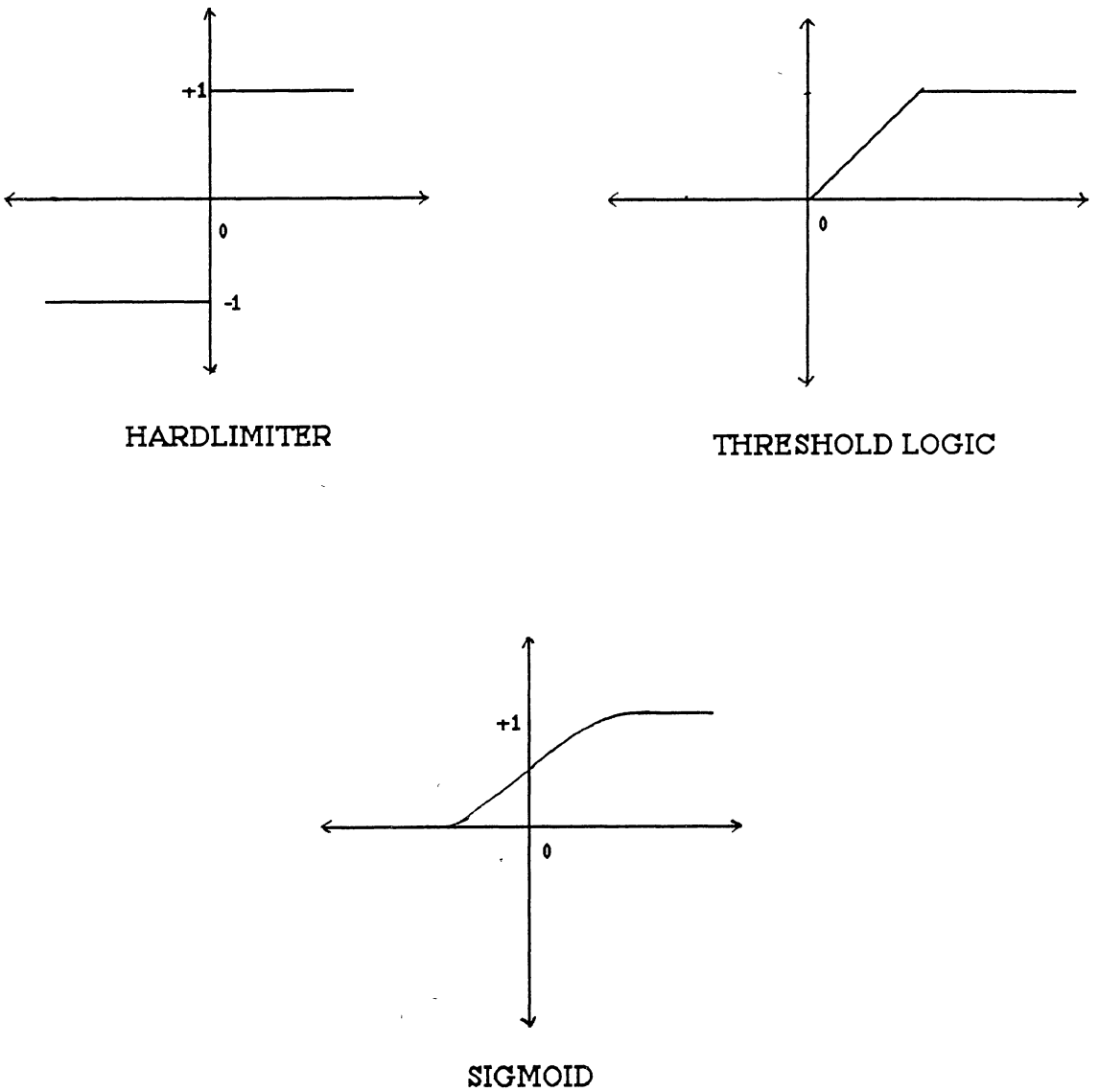


Figure 2. The Transfer Function Characteristics of Processing Elements. [LIPPMAN, 1987]

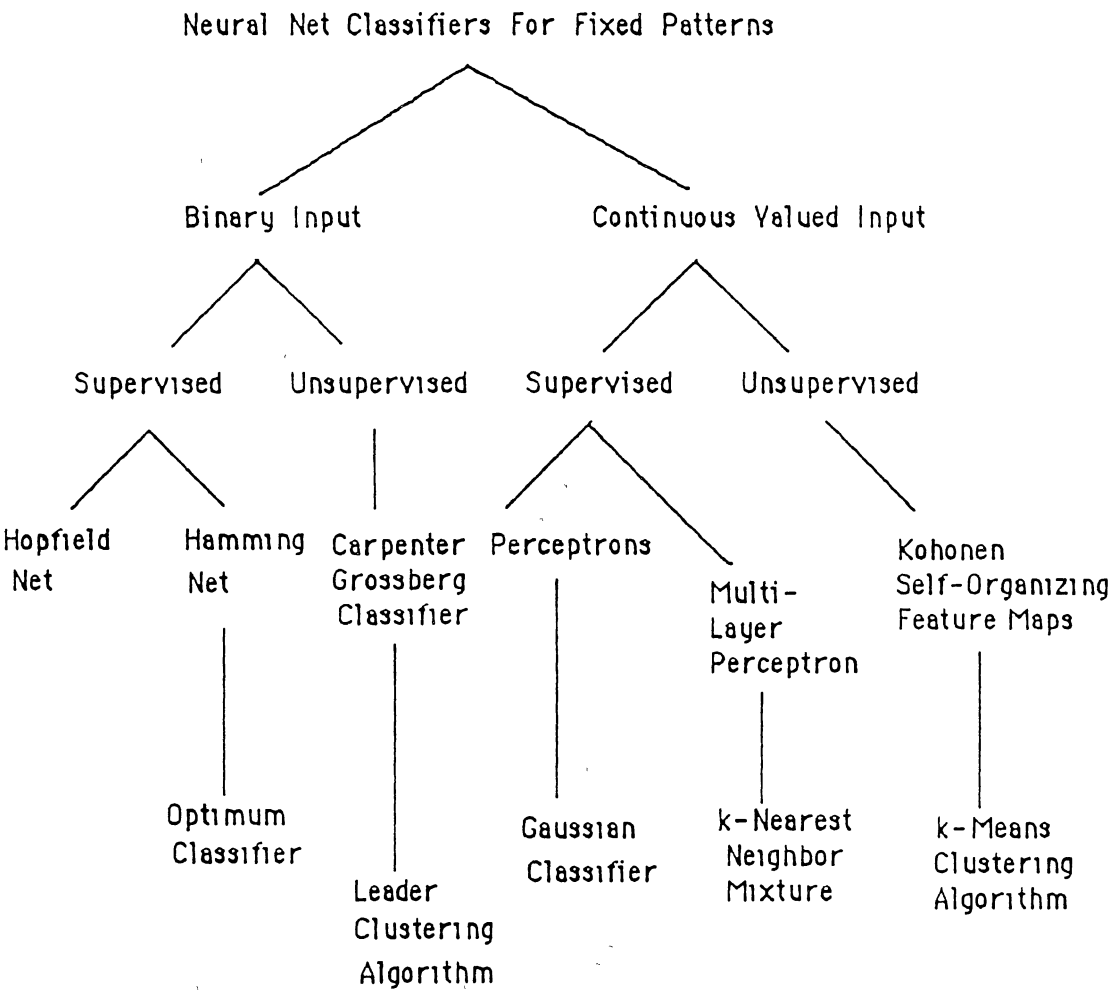


Figure 3. Classification of Neural Networks [LIPPMAN, 1987]

decisions [DUTTA and SHEKAR, 1988], image data compression [SONEHARA, et. at., 1989], bio-medical research [STUBBS, 1989], time-series forecasting [SHARDA & PATIL, 1989], pattern recognition [FALAL, 1988], and associative memories [KOSKO, 1987], and in other areas.

Prediction of time series, which is closely related to signal processing applications, using neural networks for was studied by Alan Lapedes and Robert Farber [1987]. They used a neural network with hidden layers to predict values of a nonlinear dynamical system that exhibits chaotic behavior. They used a windowing technique. Lapedes and Farber pointed out that in spite of the inability of the neural networks to provide a high degree of accuracy in numerical calculations, in this particular application the neural model gave the highest accuracy.

Backpropagation Neural Networks

Backpropagation is currently the most widely used neural network architecture. According to R. Hecht-Nielsen [1989] the backpropagation technique is defined as follows:

Backpropagation is an information processing operation that carries out the approximation of a mapping or function $f: A \subset R^n \rightarrow R^m$, from a bounded subset A of n-dimensional Euclidean space to a bounded subset f[A] of m-dimensional Euclidean space. This mapping is realized by means of

training on samples $(X_1,Y_1),(X_2,Y_2)\ldots(X_k,Y_k)\ldots$, where

$Y_k = f(X_k)$.

Further it is assumed that such examples are generated by selecting X_k vectors randomly from A in accordance with a fixed probability density function $\rho(x)$. He presents a survey of the basic theory of the backpropagation neural network architecture covering the areas of architecture design, performance measurement, function approximation, capability, and learning.

Rumelhart, Hinton & Williams [1986] describe learning internal representations by error propagation. They also present a variation of the delta rule [RUMELHART et. al, 1986] called the generalized delta rule for learning internal representations. The delta rule is a method of computing the interconnection weights that minimizes the output unit error. It forms the basis for backpropagation neural networks.

Backpropagation neural networks use a supervised learning method for training. Rumelhart et. al. explained how the generalized delta rule is a steepest descent method of computing the interconnection weights that minimize the total squared output error over a set of training vectors. In the backpropagation techniques the output generated is compared with the target vector or desired output. The differences between them constitute the error signals as

dictated by the generalized delta rule [CAUDILL, 1988]. The error in the processing element is a function of the square of the weighted vector. A plot of error versus the possible weight vectors gives a paraboloid-shaped bowl. The delta rule moves the weight vector from its current position towards the minimum error by moving down the negative gradient of the bowl. Hence it is also called the gradient-descent learning rule.

A more complicated error surface has several local minima and maxima. If the network gets struck in a local minimum, the only way to overcome this is to reinitialize the weights and retrain the network. Suggestions to improve performance and reduce the occurrence of local minima include allowing extra hidden units [LIPPMAN, 1987]. One other difficulty than the local minima problem is that often this technique requires a large set of training data for convergence. Typically more than 100 passes may be required to make the network converge to a minimum error region. Even though several more complex adaptation algorithms have been proposed it is very unlikely that complex decision regions formed by multi-layered networks can be generated in fewer trials [LIPPMAN, 1987]. This is especially true when the decision regions are disconnected. In spite of these drawbacks, backpropagation has been proved to be successful in many applications.

Backpropagation neural networks always form hierarchical structures, in the sense that these networks always have at least three layers of neurons. Werbos [1989] highlights the emerging field of neurocontrol, and describes how backpropagation can be used in a new generation of controllers suitable for problems involving thousands or millions of control variables and a high degree of noise and nonlinearity.

Chauvin presented a variation of the backpropagation algorithm that makes use of network hidden units in a optimal manner [CHAUVIN, 1989]. The algorithm he proposed automatically finds optimal or nearly optimal architectures necessary to solve known Boolean functions. It also facilitates the interpretation of the activation of the remaining hidden units. He explains how the network can be used in realizing complex signal processing algorithms.

Signal Processing

Recovering transmitted analog signals over a noisy or dispersive channel is an example of classical signal processing problem. Yeates [1989] describes a neural network structure as well as its ability to carry out several classical signal processing algorithms. He also describes an important filtering technique, Kalman filtering, for a general state transition matrix. Kalman

filtering is done by allowing weight changes to depend, locally, on other weights in the proposed model. He develops a novel parallel architecture for signal processing which is of interest for two reasons. The structure he proposes exhibits characteristics similar to the proposed schemes used in neural modeling. The architecture implements a Quasi-Newton algorithm with superior convergence properties to that of the Least Mean Square (LMS) algorithm implemented by the adaptive digital filters.

The noise present in a signal can be of two types - stationary or random. The stationary disturbances can be eliminated by adaptive prediction. Prediction is concerned with the problem of extrapolating a given time series into the future. As for filtering, if the underlying model of the time series is known, it is possible, in principle, to design an optimal predictor for future values. When the model of the time series is not specified, it is plausible that a model could be estimated by analyzing past data from the time series [GOODWIN & SIN, 1985]. Random disturbances are models of stochastic processes. If the signal and noise spectra are non-overlapping, then it is possible to design a filter that passes the desired signal but attenuates the unwanted noise component. The resulting filter would be of low-pass, band-pass, or high-pass type, depending on the relative frequencies of the desired signal and noise. There

are standard procedures for design of such electrical filters; these are discussed in OPPENHEIMER & SCHAFER [1975], OPPENHEIMER [1978], and RABINER & SCHAFER [1980].

The case of overlapping signal and noise spectra was first studied by Weiner and Kolmorgorov [1948], who formulated the filter design problem using statistical and frequency domain ideas. The usual method of identifying a signal embedded in additive noise is to pass the composite signal through a filter that suppresses the noise while leaving the signal relatively unchanged.

Radar signal processing with multi-layered neural networks was described by Solka, et. al., [SOLKA et. al., 1989]. They tested networks with single hidden layers on millimeter wave target returns which had been corrupted with Gaussian noise. Speech production using a neural network with a cooperative learning mechanism was developed by Komura and Tanaka [1987]. Their proposed model consists of four layers including hidden and multiple outputs. They succeeded in producing natural speech waves with high accuracy.

Various digital filters used in signal processing applications are described by Jackson [JACKSON 89]. He points out that the wealth of analog filter design data and techniques that has evolved over the years is based on frequency domain specification. For this reason, many

design techniques for infinite impulse response (IIR) digital filters utilizing analog filter design have been developed. The transformations from the analog variable in the Laplace transform domain to the digital variable using Z-transforms have been derived. The pioneering work in this area was done by Kaiser [KAISER, 1973].

Johnson [1989] presented the theory and design principles involved in the design of finite impulse response (FIR) and infinite impulse response filters (IIR). He discussed in depth the design techniques of FIR filters based on windowing, frequency sampling, and optimization. The windowing method involves straight-forward analytical procedure. To obtain a realizable filter, the impulse response sequence is truncated to obtain a new sequence having finite domain. This truncation process is called windowing. The Kaiser family of windows provides the designer considerable flexibility in meeting the filter specifications. In the frequency sampling method, a desired frequency response is uniformly sampled and filter coefficients are then determined from these samples using discrete Fourier transforms. Optimal FIR filters have well-defined analytical procedures and solutions that are used in the design of these filters. The optimal filters frequently are based on Chebyshev approximation.

The adaptive linear combiner or non-recursive adaptive

filter is fundamental to signal processing applications. The general form of the adaptive linear filter is shown in Figure 4. The adaptive linear combiner takes in a signal vector and has a corresponding set of adjustable weight vectors and a summing unit to form the output signal. The method of adjusting or adapting the weights is called "adaptation" procedure. The combiner's fixed weight settings causes the output to be a linear combination of input components.

A general digital signal processing system is shown in Figure 5. By setting the feedback coefficients to zero, we get the single-input adaptive combiner as shown in Figure 4. The system shown in the figure is also called a digital filter. Without the feedback portion, the filter is called "non-recursive" and with the feedback portion it is "recursive." The non-recursive digital filters such as the adaptive linear combiner are inherently "stable." This can be attributed to the weight values. As long as weights are finite, the impulse response is bounded and is of finite length. Such digital filters are called Finite Impulse Response filters (FIR). The recursive filters have impulse responses of infinite lengths. These filters are called Infinite Impulse Response (IIR) filters.

The adaptive noise canceling technique was first studied in depth by Widrow in early '60s. Widrow et. al.

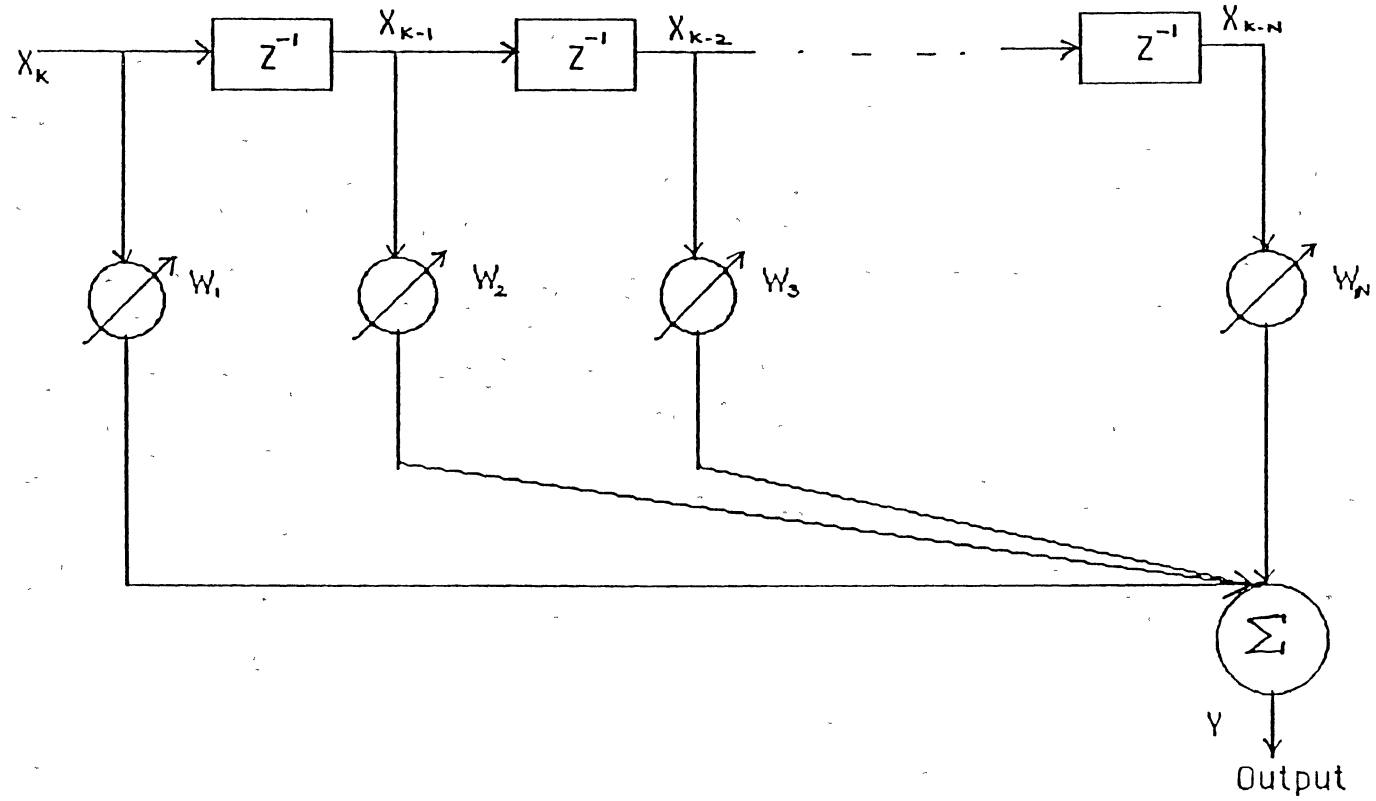


Figure 4 . Single Input Adaptive Combiner [WIDROW et. al., 1985]

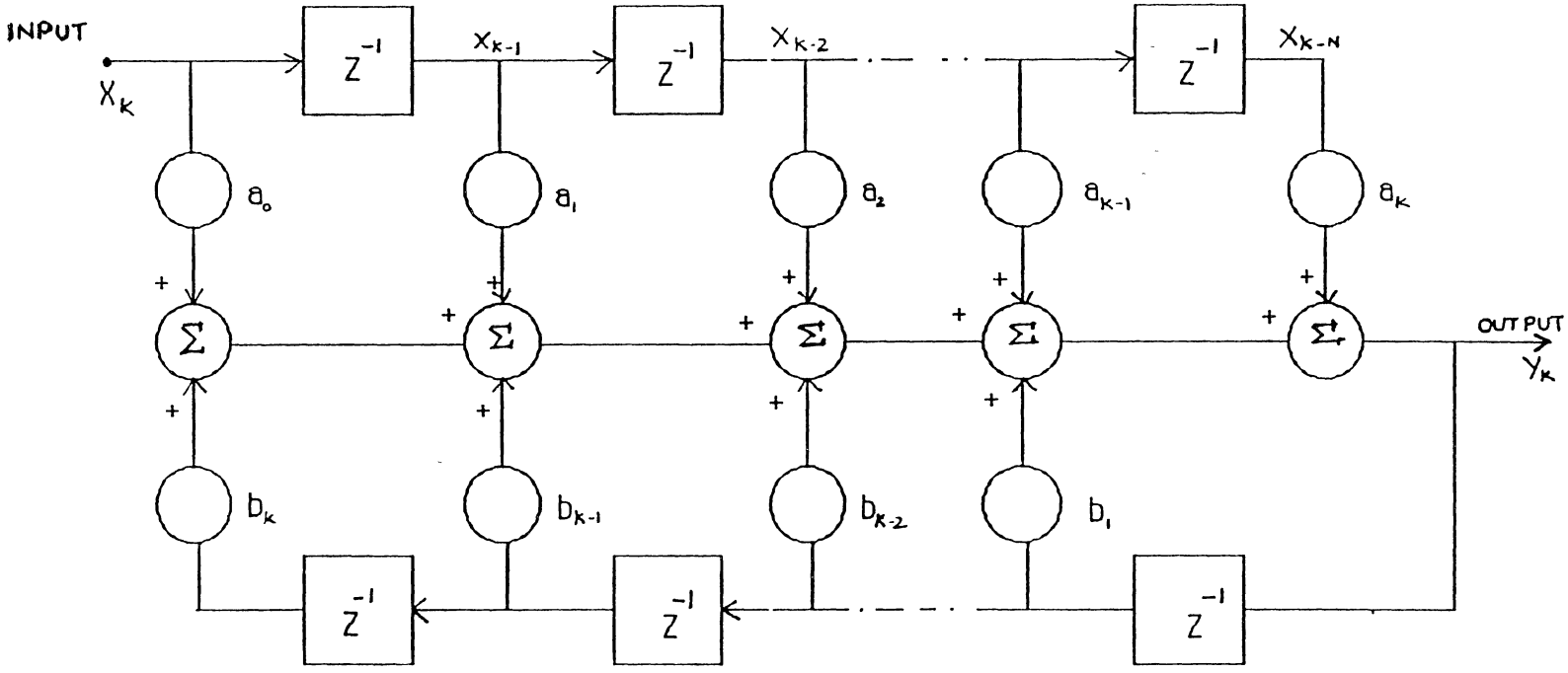


Figure 5. Digital Signal Processing System [WIDROW et. al., 1985]

[1975] presented principles and applications for adaptive interference canceling and explained how the adaptive filtering technique can be applied successfully for EKG signal processing. Their model was closely based on the least mean square rule (LMS), which later formed the basis for the delta rule.

In 1960, Widrow and Hoff presented a learning rule called delta rule, which forms the basis for neural networks with hidden units. In fact, there is no equally powerful rule for learning in networks with one or more hidden units. A slight variation of this rule, called the generalized delta rule, is used by most of the backpropagation neural models with hidden units known today.

Huhta and Webster [1973] pointed out that a major problem in the recording of the electrocardiograms (EKG) is "the appearance of unwanted 60-Hz interference in the output" [WIDROW et. al, 1985]. They analyze the various causes of such power-line interference, including magnetic induction, displacement currents in leads or in the body of the patient, and equipment interconnection and imperfections. They also describe several techniques for minimizing interferences which must be used during the recording process itself, such as proper grounding and use of twisted pairs. Another method capable of reducing 60-Hz ECG interference is adaptive noise cancelling, which can be

used separately or with more conventional approaches [WIDROW and STEARNS, 1985]. The standard approach to this problem is to adapt the finite impulse response filter (FIR) to reduce the noise and to use channel equalization to reduce the dispersion, and to apply estimation theory to estimate the optimal desired waveform. Neural networks is a powerful adaptive technique providing solutions to signal processing or signal to noise ratio reduction problems [MELNIKOF, 1989]. One of the particularly interesting characteristics of neural networks is the capability to develop an adaptive filtering technique that can be tuned to preserve varying levels of details [KLIMASAUSKAS, 1989].

EKG processing demands high speed and accuracy. Data gathered from patients with suspected heart ailments typically approaches 100,000 or more beats over a 24 hour period within which there may be only one or more sequences of abnormal beats [CAROLL and VED, 1989]. They propose a neural network based model for identifying this arrhythmia from a noise free signal. It is very important that these signals are identified clearly without any noise for proper diagnostics. A data compressing algorithm for Digital Holter Recording using artificial neural networks (ANN) was developed by [IWATA, NAGASAKA, SUZUMARA, 1989]. They describe a neural network system for monitoring EKG signals for detecting disorders and arrhythmias in EKG which may

appear unusual or non-periodically. Goodwin et. al. [1984] explain how adaptive filtering prediction and control can potentially be applied to filter the noise component from the EKG signals.

The neural network filter can be useful in the signal processing applications which require high speed, accuracy, and adaptability to the changing input signal with time. The main process involved in these filters is in compressing or encoding the input data and re-expanding it. The compression process eliminates that part of the input signal which represents small or nonrecurring features. By varying the number of encoders the actual detail in the output signal can be varied.

Figure 6 represents the process discussed above. The main step in solving this problem is the selection of a good set of feature detectors by which the particular signal can be filtered. In the figure shown, there are 14 inputs which have been reduced at the output encoder to 4 outputs. Representing these inputs and outputs as 32 bit floating point numbers requires (14x32) and (4x32) bits to represent the input and output patterns respectively. From this viewpoint, the data has been reduced by a factor of 448/128. This form of compressing the data is called dimensionality reduction [KLIMASAUSKAS, 1989].

The compressed signal from the hidden layer is re-

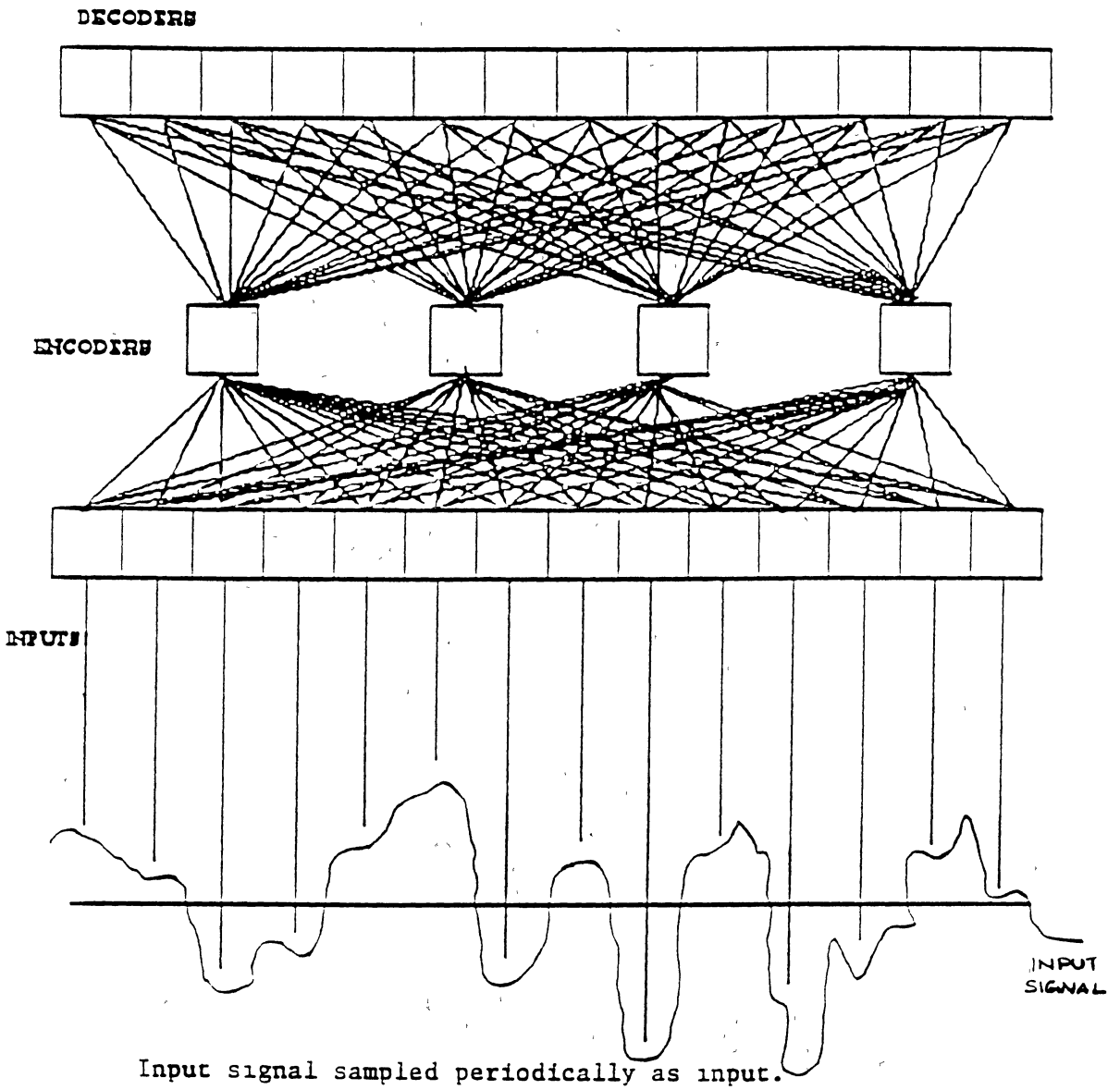


Figure 6. Dimentionality Reduction. An Input Signal is Sampled at Several Points and Encoded. The Decoders Reconstruct the Full Input Signal. [KLIMASAUSKAS, 1989]

expanded through the connection between the hidden and output layer. The backpropagation neural networks configure themselves in such a way as to establish a relationship between the input and the output variables. The connection is formed to minimize the error between the network generated output and the target vector. After minimizing the error or training the network, the network can be said to have learned the input/output relation. Such a network produces an output close to or consistent with the target pattern, even when one or more inputs have been corrupted. This is attributed to the non-linear characteristics of the neural networks.

Signal processing using neural networks is an ongoing research topic. This study verifies the ability of neural network techniques in noise filtering and compares it with the performance of a finite impulse response filter (FIR). The noise in the signals is stochastic. This requires complicated nonlinear signal processing algorithms whose procedures are less known than stationary noise signal processing algorithms. A neural network, by its ability to change its structure according to the environment, can be used in nonlinear signal processing which requires adaptability. One of the important features of neural networks is the ability to develop adaptive filtering techniques which can be adjusted to identify varying levels

of detail in the underlying signal. The output of the FIR filter and neural network based filter is compared in terms of signal reconstruction and distortion of the output waveforms. The input signal pattern which corresponds to the duration of one heartbeat is extracted from the original time series. The network first is tuned with the supervised signals which are same as input signals. About 300 successive heartbeats is used for training the network. After the tuning up, the activations of the neurons correspond with the characteristics of the waveforms. A recall file containing the unfiltered signal pattern is then presented to test the network's ability to filter the noise and identify the waveforms.

CHAPTER III

IMPLEMENTATION OF NEURAL NETWORK FILTER

Training Neural Networks

The neurons in a neural model can be "feedforward only" or "feedback only" or a combination of these two. Training in neural networks can be classified into three categories: supervised training, unsupervised training, and self-supervised training. In the present application, the backpropagation neural model, which uses supervised training, is used. In supervised training mode, the neural model is capable of discriminating among the members of a set of stimulus inputs, by providing the network with the correct response expected at the output. An external signal corrects the network to produce the desired response.

Unsupervised learning uses the inputs and forms appropriate interconnections to produce the desired response. No external input is provided to produce this response. One common unsupervised learning procedure is competitive learning. In this technique, the inputs compete among themselves to produce a response to each applied stimulus pattern. The winner is the one whose connection

made it respond most strongly to the pattern. The network adjusts its interconnection strengths slightly towards the pattern that won. This technique is mostly used in classification problems. Much of the current research on the unsupervised learning scheme is focused on the emergence of feature detectors. Several researchers, including Rumelhart and Zipser [1986], built a network that can discriminate between stimuli occurring on spatially different parts of an input visual field using a form of competitive learning. While unsupervised learning schemes are topics for on going research, they are not as well understood as supervised learning techniques. This has mostly precluded incorporating unsupervised neural models into useful applications.

In self-supervised learning, the network model is presented with a series of input training patterns. The network learns to classify these patterns without being given the correct response for an input pattern. An error signal generated by the network is fedback to itself and the network adjusts its interconnection weights to produce the desired response. The network produces correct response after several training cycles. Networks trained with supervision include perceptrons and multi-layer perceptrons.

Training Using Backpropagation

Neural Networks

The neural network filter developed in this study uses supervised learning and in particular the backpropagation technique. Backpropagation is the most widely used learning technique. Backpropagation learning techniques use the Delta or least mean square (LMS) rule to adjust the weights on the interconnections to produce the correct response. Earlier, when neural network models were formed, they consisted of simple two-layer associative networks in which a set of input patterns applied to an input layer was mapped to a set of output patterns by the output layer. These models involve only input and output and do not have internal representation. The main disadvantage of such systems is the inability of the network to learn certain mappings of relatively low complexity. The exclusive OR (XOR) problem is a very good example.

Minsky and Papert [1969] showed several non-linear problems which two-layer networks did not solve. Their study showed that it is possible to do mapping required to solve problems that are not linearly separable. Later it was proved that a layer of simple neurons between the input and output layer will solve the problem. The third layer is called the hidden layer. The learning in backpropagation is

closely based on the Delta rule. The delta rule was developed by Widrow and Hoff [1960] and recently has been reanalyzed and studied by several authors. The delta rule is also called the Widrow-Hoff rule.

Generalized Delta Rule

The learning in neural networks during training is derived by applying a pattern to the network and allowing the activations to be passed to the output layer through the hidden layers. The weights are modified layer by layer by propagating the difference of the comparison of the output pattern to the target pattern. The error, also called delta, is propagated to the previous layer and while it propagates it modifies weights. The learning ceases when the error signal becomes zero.

The rule for changing weights given by Rumelhart, et. al. [1986], for a set of input and output pattern p is

$$\Delta_p W_{ij} = \eta (t_{pj} - O_{pj})\ i_{pi} \quad (1)$$

$$= \eta\ \delta_{pj}\ i_{pi} \quad (2)$$

where

t_{pj} is the target input for the jth component of the output pattern for pattern set p.

O_{pj} is the jth element of the actual output pattern for pattern set p.

i_{pi} is the value applied to the ith element

of the input pattern p.

$\delta_{pj} = t_{pj} - O_{pj}$ is the actual error produced by comparing the target output with the actual output.

$\Delta_p W_{ij}$ is the actual change in weight magnitude that is to be made after pattern p is applied. (W_{ij} is the weight vector; the strength of the connection from unit i to j).

The informal derivation of the delta rule as given by Rumelhart et. at. [1986], is given in appendix A. The derivation shows that the net change in weight values, W_{ij}, after a cycle of input patterns, is related directly to the derivative of error with respect to weight. Hence the delta rule always moves the weight vector in the direction of negative slope to reduce the overall error, E.

Internal Representation in Backpropagation Neural Networks

The internal details of the backpropagation neural model are shown in Figure 7. The first layer or the input layer consists of processing elements or neurons which fan out to 'n' processing elements. The elements in this layer take in individual components of the input vector. These are distributed to all elements in the second layer (hidden layer) without any changes. Each element in every layer

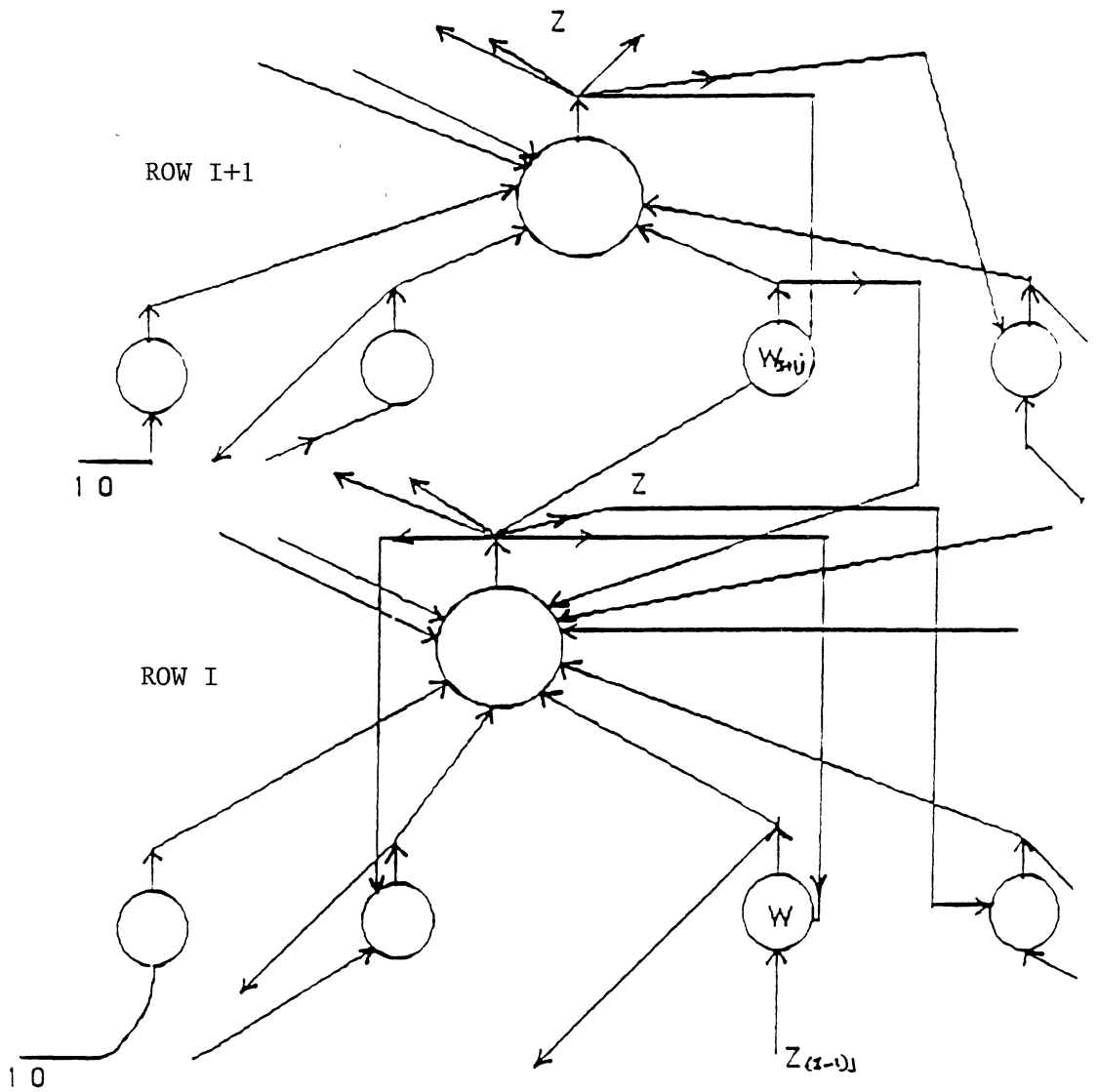


Figure 7. Internal Details Of Backpropagation Networks
[R. HECHT-NIELSEN, 1989]

except the input layer receives the output of the units in the layer immediately below it. The final layer, also called the output layer, accepts input from the layer immediately below it and tries to produce the desired output, the network's estimate of the output pattern. Any layer between the input and the output layer is a hidden layer. Theoretically there can be any number of hidden layers in a neural model.

The backpropagation neural networks function in two stages: the forward pass and backward pass. During the forward pass stage, the signals propagate from the input layer to the output layer through one or more hidden layers. Once the output has been estimated by the forward pass, each of the elements in the output layer is given its component of the correct output pattern, the target. This initiates the backward pass. In the backward pass stage, the error signal is computed by taking the difference of the network's estimate of the output component and the target component. This error signal is passed back starting from the output layer. This backpropagation of the error signals adjusts the weights depending on the error signals, so that the network produces the correct response. Each cycle of training the network involves "bubbling up" the inputs from the output layer to the input layer, while the error "propagates down" from the output layer to the input layer.

The forward pass or backward pass is determined by the scheduling processing element. The scheduling processing element sends signals to each processing element of the network telling it to apply its processing elements transfer function and whether to apply forward pass or backward pass propagation.

Each processing element in the backpropagation neural model consists of one "sun" processing element and several "planet" processing elements. The sun of the processing element gets input signals from the planets which go to the suns of the previous layer as well. In turn the planet receives input from its sun and also from the previous layer. A detailed description of working of backpropagation networks is given by Hecht-Nielsen [1989].

Error Surfaces in Backpropagation Neural Networks

The error surface of a backpropagation neural network can be defined by the equation [HECHT-NIELSEN, 1989] as:

$$F_s = F(W) \tag{3}$$

in the Q+1 dimensional space of vectors (W,F).

W is the weight vector of the network.

Q is the number of dimensions in vector W. (i.e. planets in the network)

For each W a non-negative surface height F_s is defined by

F(W). (The weight vector, W, ranges from over its Q-dimensional space). While approximating the function, f, the network makes an average squared error F(W), for any selection of weights W. F(W) is defined to be

$$F(W) = Lt\ (1/N)\ \Sigma_k\ F_k \qquad (4)$$

where

$$F_k = \{f(X_k) - B(X_k,W)\}^2 \qquad (5)$$

f : is the given mapping function

k : is the number of the testing pattern

B : is the network estimate of the output which is a function of input vector, X, and network weight vector, W, i.e. $B(X_k,W)$

N : is the total number of patterns.

The function F(W) is the mean squared error function of the network.

The generalized delta rule used by the backpropagation neural network moves the weight vector, W, from the starting point w_o to reduce the mean squared error. There is a zero probability that w_o is already a minimum. The backpropagation error surface consists of flat regions and troughs that have very little slope, so the weight vector must be moved quite a long distance on the error surface before F(W) drops significantly. Due to the small slope the delta rule has a hard time in determining which direction to move the weight vector to reduce the overall error.

The local minima in backpropagation error surface were discovered recently [HECHT-NIELSEN, 1989]. When the network gets struck in a local minimum, the weight vector has to be moved away from it to find a global minimum. The training has to be repeated after perturbing the weights.

Training Samples and Environment

EKG signals are used for implementing and testing the simulated neural network based filter. Clinically recorded electrocardiograms formed the training samples. The EKG signal recorded by the EKG recorder is a continuous stream of binary numbers. These signals are first converted into ASCII numbers using programs developed in Pascal. The text file is then converted into a sample vector corresponding to a heart beat. Approximately 110-115 samples constitute one heart beat. About 250 such heart beat patterns are used to train the neural network. These patterns form the input vector. A part of the noisy signal is applied to the FIR digital filter and the filtered output obtained is used for the target signal vector, after suitable conversion and formatting. The training file consists of pairs of input pattern and the target pattern.

Simulation Using Artificial Neural Simulation Tool

The neural network filter was simulated using Science Applications International Corp.'s (SAIC) ANSim, Artificial Neural Simulation Software. ANSim is a graphics-oriented, menu-based, artificial neural network simulation program running on top of Microsoft Windows. ANSim supports 13 presently well known neural network paradigms.

Figure 8 summarizes the steps involved in artificial neural network system development and application. This gives only a general idea of the process; particular applications may involve repeating or excluding one or more steps or may have steps which are more complex and compounded. For example, in some cases the difference between training and processing is non-existent. Some models need no training at all and in others training is the same as processing except the use of a special training data set. The network models developed using ANSim can be embedded in stand-alone applications.

The backpropagation paradigm uses training data files containing sequences of vector pairs for training the network. Each vector pair is composed of an input vector and a target vector. The number of components in the input vector is equal to the number of elements in the input layer

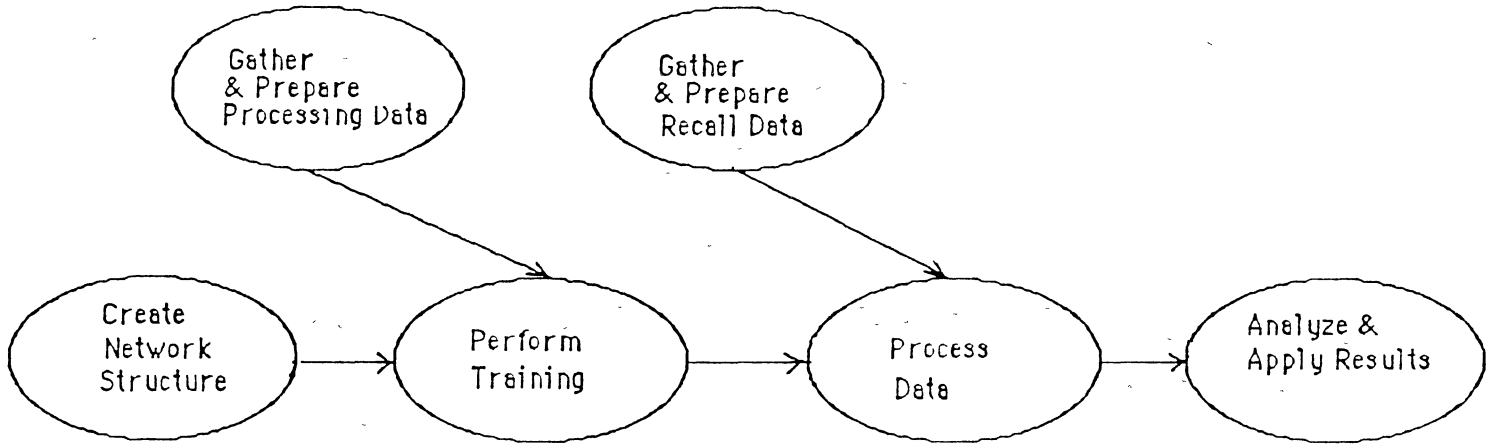


Figure 8. Steps Involved in Development of Artificial Neural System

of the network model. Similarly, the number of components in the target vector must equal the number of neurons in the output layer of the network. The recall file contains sequences of vectors for driving a trained network. The recall file does not contain target vectors. The components in each vector are the input values applied simultaneously to the network. The output produced by the network model while processing the recall file have the same structure as recall files so that they can be used as input to other networks. The training or recall files should be normalized before presenting to the network. Normalization is done to remove the mean and to scale data within a specified range. During scaling, all values are adjusted proportionally so their minimum and maximum fall within the specified range. The backpropagation network represented in ANSim is a multilayered, fully connected, feed-forward network. It is based on the Generalized Delta Rule (GDR).

CHAPTER IV

RESULTS AND DISCUSSION

Backpropagation neural network systems are a powerful adaptive system for forming relationships between several valued continuous inputs and continuous valued outputs. The interesting property in neural network filters is their ability in identifying the noisy and corrupt input patterns. This can be attributed to the non-linear mapping ability of neural networks. To illustrate the process in a signal processing application, Figure 9 shows the adaptive interference canceling of EKG signals using LMS algorithm. The input and target wave samples used in training the network are shown in Figure 10. The input signal is corrupted by random noise. The target pattern is the filtered output obtained from the FIR filter. The network attempts to produce output similar to the target pattern.

The output produced by the neural network filter for various number of elements in the hidden layer is shown in the Figure 11. It can be seen from the figure that, the more the number of processing elements in the hidden layer, greater is the detail preserved from the encoding process to decoding process. However, the amount of detail produced at

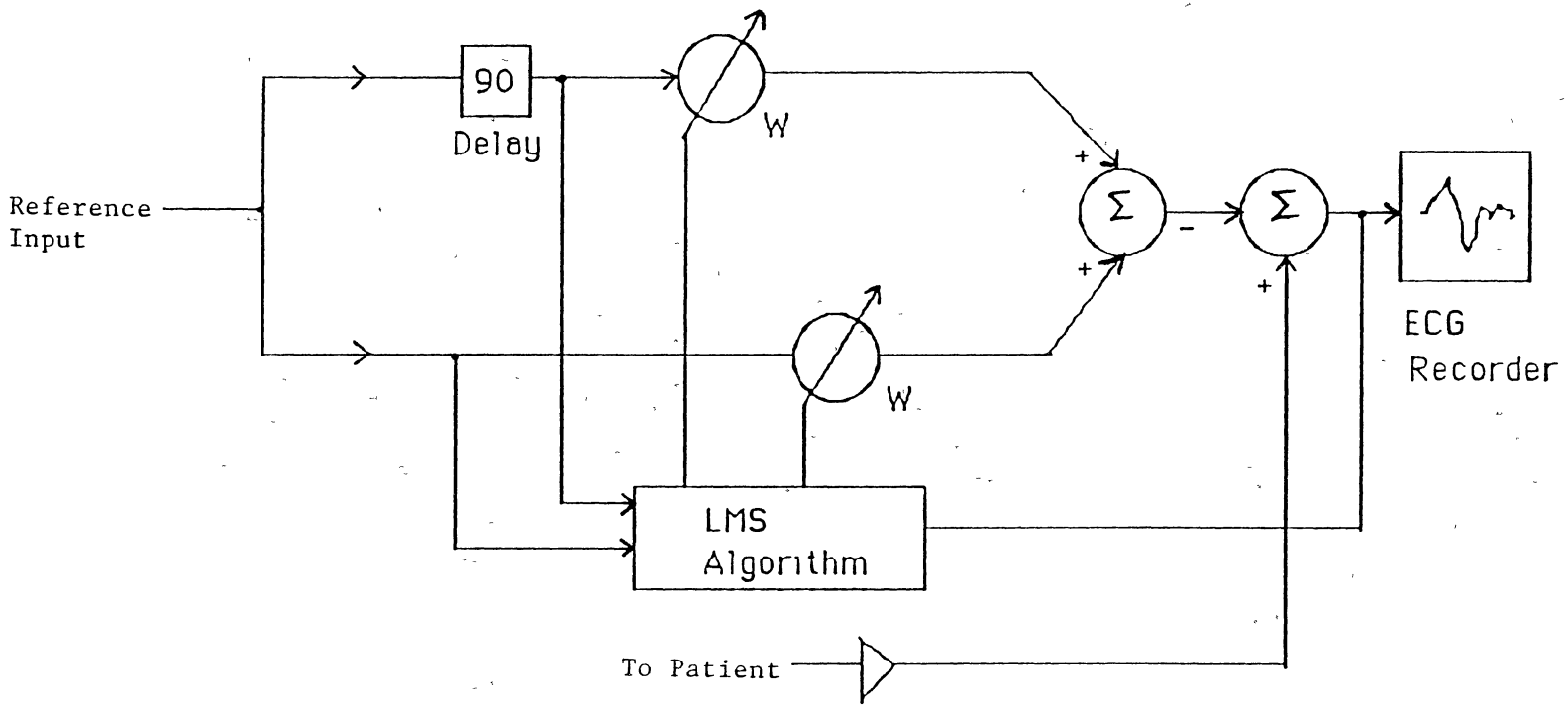
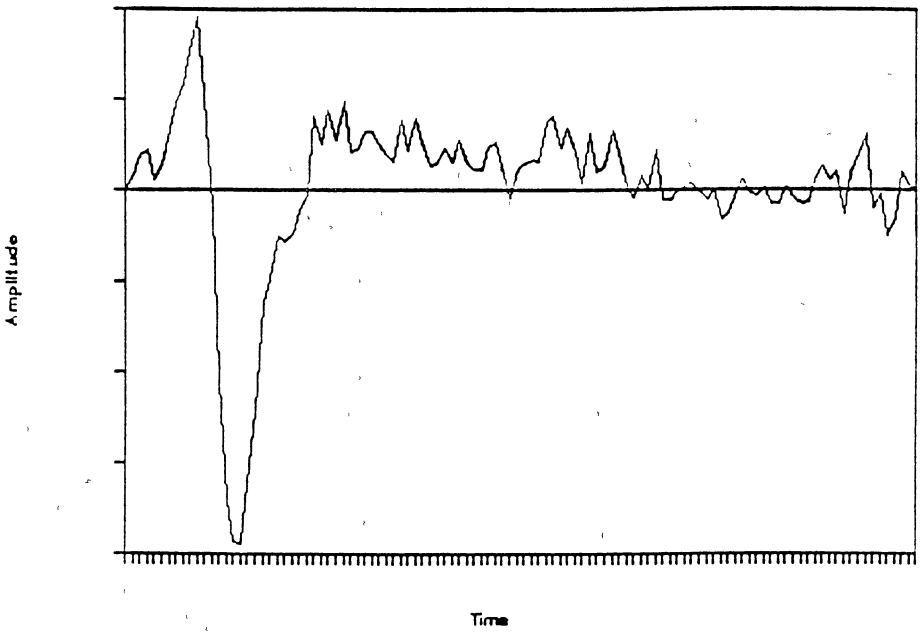
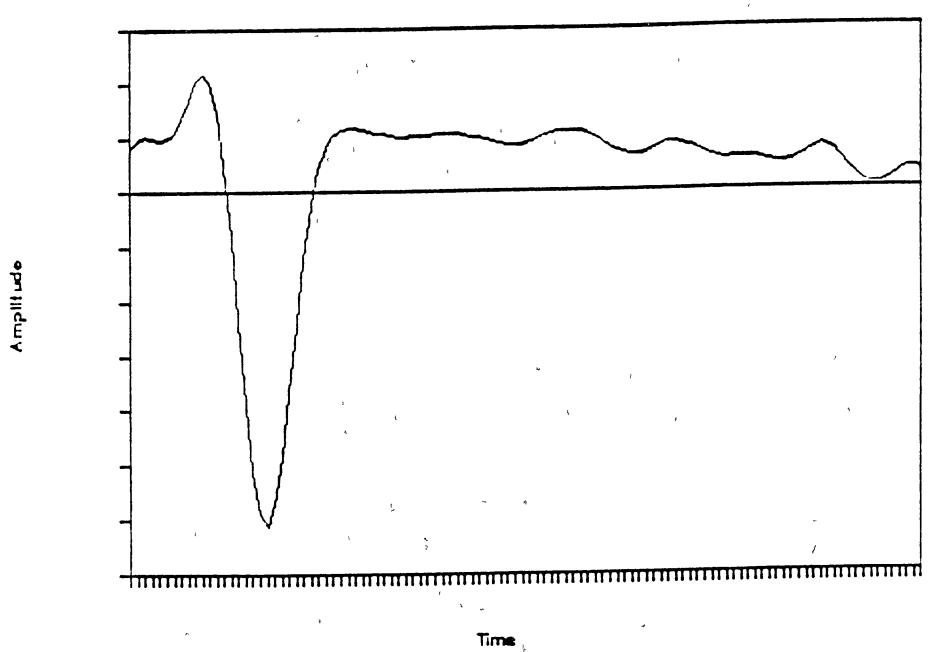


Figure 9. Adaptive Noise Cancelling in EKG Signals [WIDROW et. al., 1985]



Input Signal Pattern to the Network Filter.



Target Pattern to the Network Filter

Figure 10. Training Input and Target Patterns.

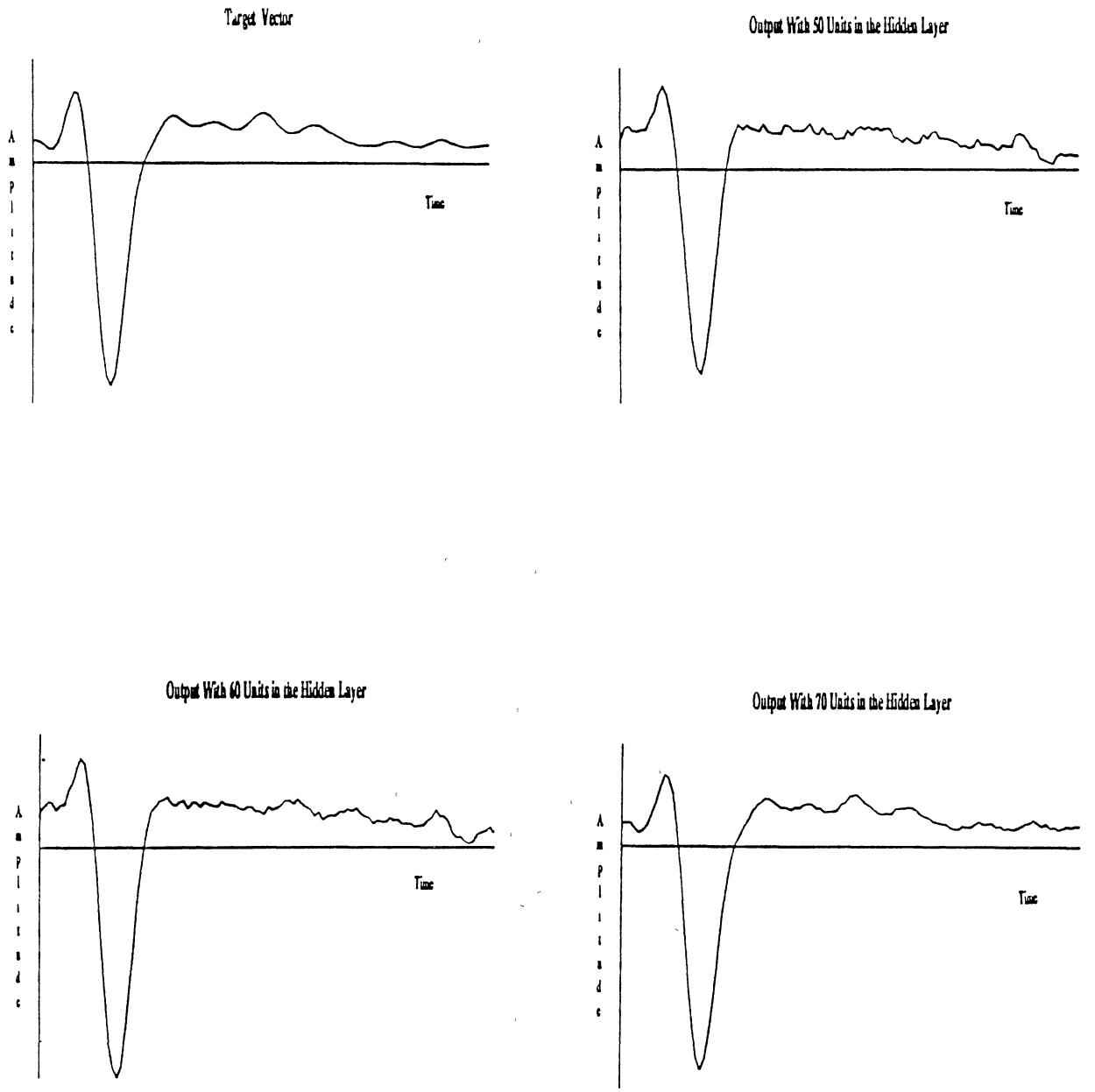
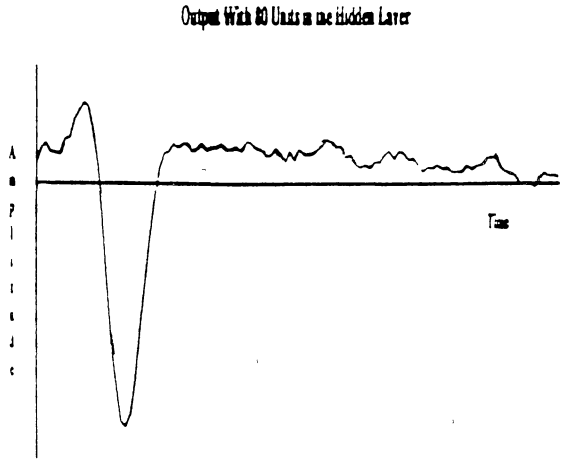
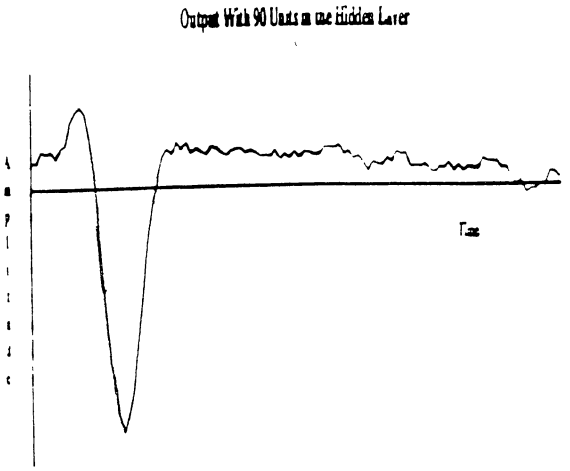
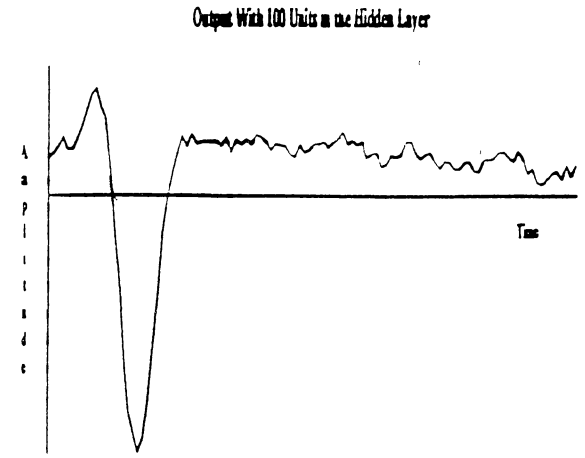


Figure 11 . Output of Network For Various Number of Units in the Hidden Layer







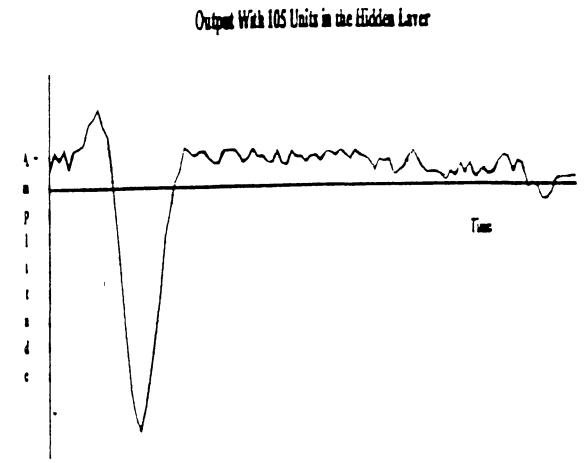


Figure 11. (Contd.)

the output layer does not improve after 70 processing elements in the hidden layer. The performance of the network actually starts to degrade slowly after this stage. When the hidden layer had about 105 units the network output was almost the input pattern.

Thus smoothing of the corrupt and noisy signal input is determined to a great extent by the number of units in the hidden layer. Usually, the number of units in the hidden layer is kept well below the number of units in the input layer below it. The lesser number of units in the hidden layer guarantees smoothing of the input signal. Figure 12 shows the effect of the number of elements in the hidden layer on the output unit error for 110 elements in the input layer.

The performance of the neural network filter during training increased as the training progressed. However, the continued training after sometime does not improve performance. The root mean square error and maximum unit error does not reduce significantly after this stage.

Figure 13 shows the maximum output unit error with respect to time (training cycles). It was observed that during training the maximum output unit error decreased steadily initially. After about 250 cycles the decrease in error at the output unit becomes insignificant. At this stage it can be said that the system has learned. Further

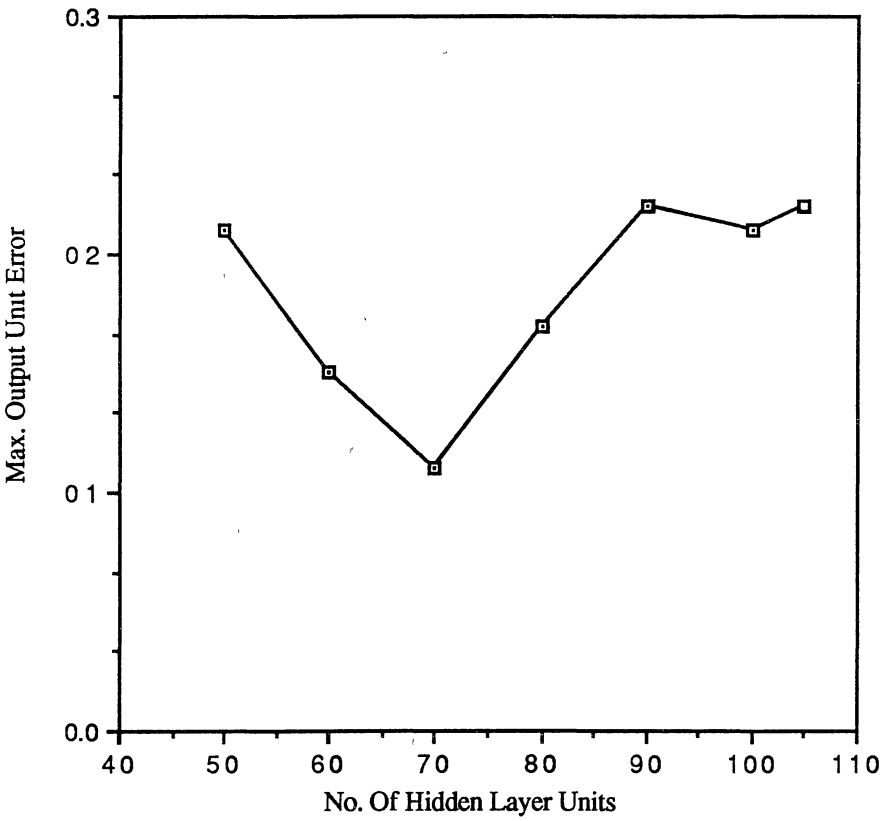


Figure 12. The Graph Shows the Effect of Hidden Layer Elements on the Output Unit Error.

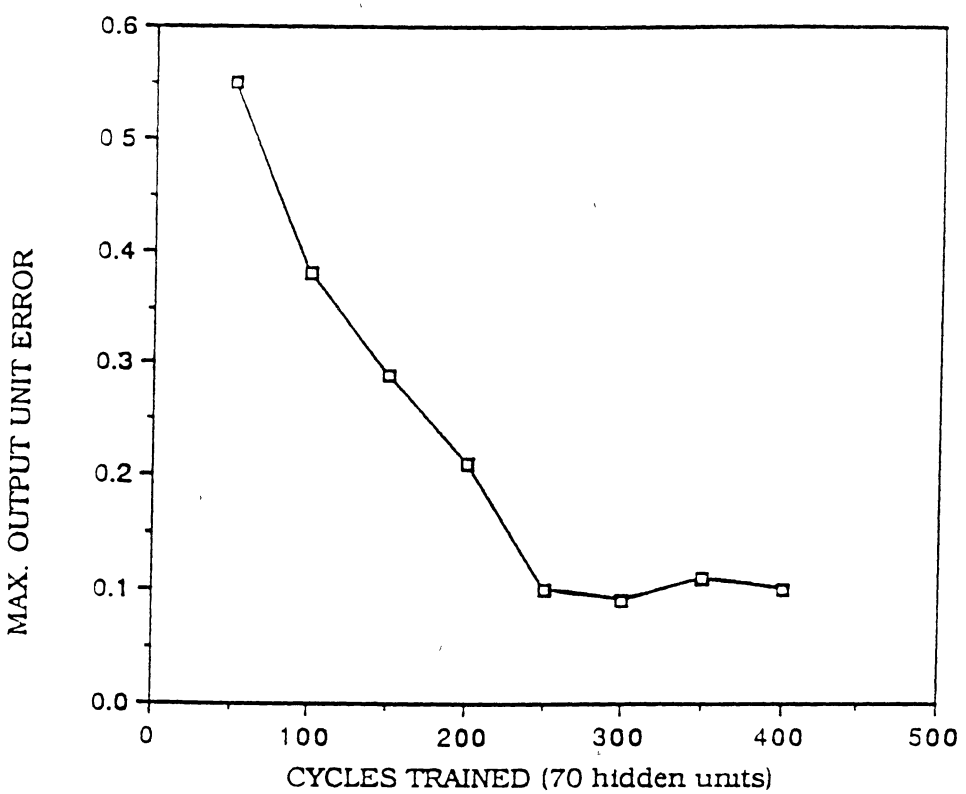


Figure 13. Training Curve. The Graph Shows the Effect of Number of Training Cycles on Output Unit Error.

increase in training cycles does not reduce the output unit error significantly.

It was also noted that performance on the testing data increased with the number of training samples. However, if training samples were increased beyond 40, the network performance did not continue to improve. The increase in the number of samples during training makes the network to learn for a range of patterns. This improves the performance of the network when applied with the testing set. Figure 14 shows the effect of training vector size on the output unit error.

The learning rate (η) influences learning to a great extent. The goal is to set the learning rate as high as possible without causing the output error to oscillate. The optimal value of η depends on the shape of the error function in weight space. Figure 15 shows the output unit error against the learning rate. When the learning rate is very low, the network output error oscillates and does not reduce. On the other hand a high learning rate also does not reduce the output unit error. Difficult problems have relatively constant error functions with tiny solution regions. These problems require a small value of η, typically 0.25 or less, and require many learning cycles. In practice, the value of η should be reduced until the output unit error shows a decreasing value with time.

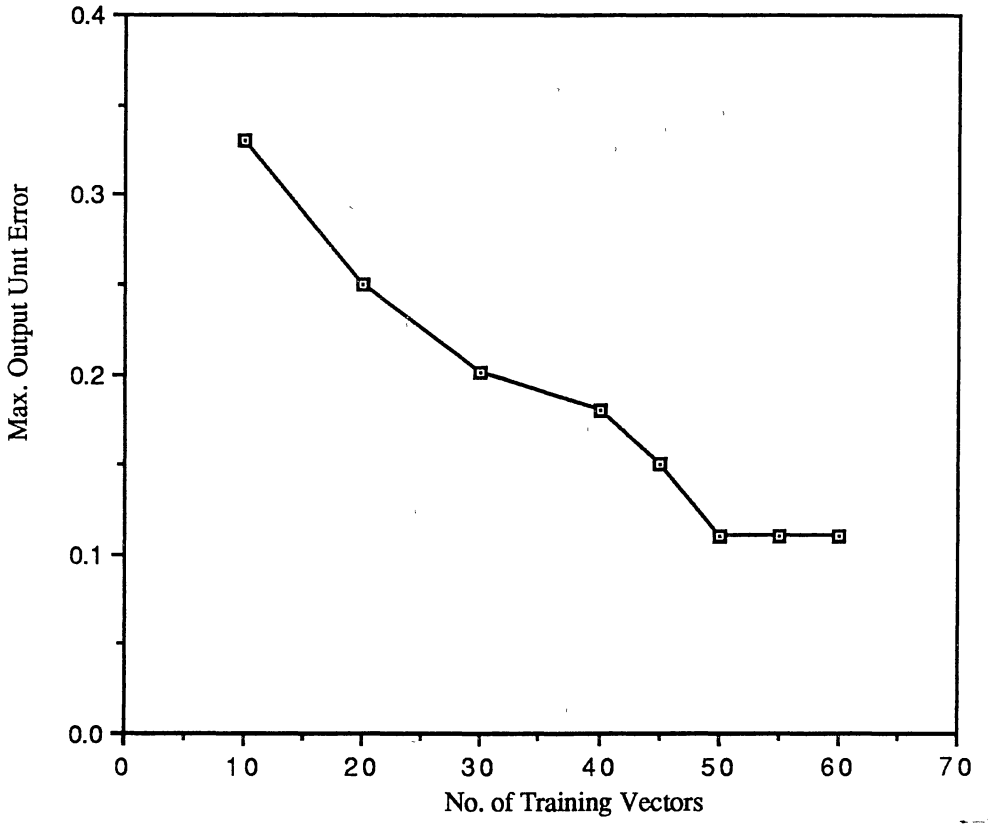


Figure 14. The Graph Shows Output Unit Error For Various Number Of Training Vectors.

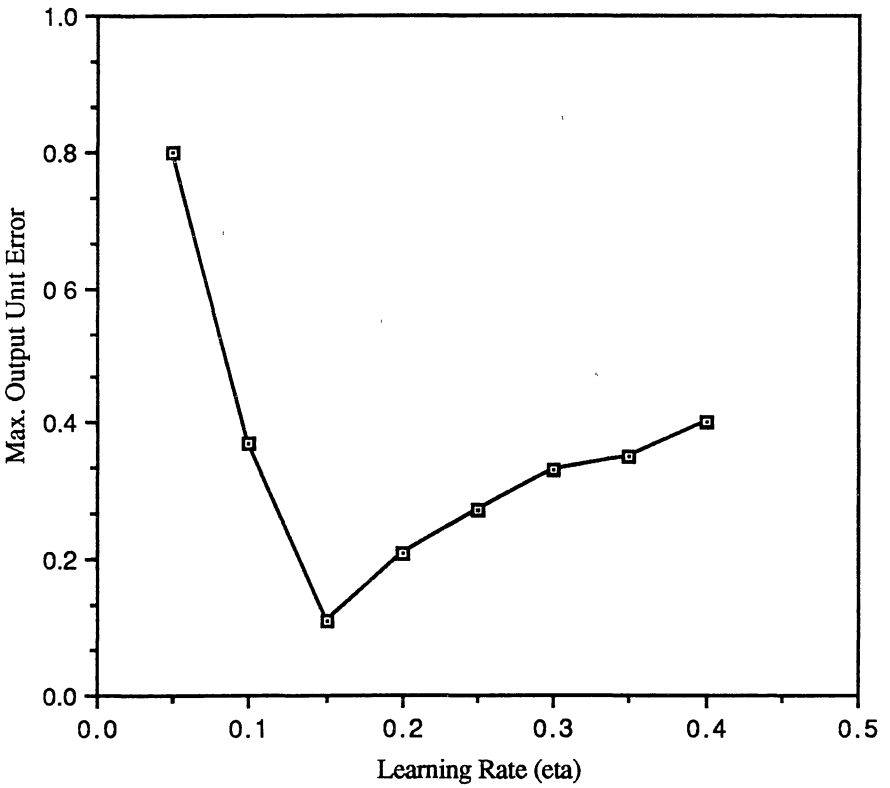


Figure 15. Learning Curve. The Graph Shows the Effect of Learning Rate on Output Unit Error.

Klimasauskas [1989] showed how neural networks can be applied to noise filtering successfully. He showed how the hidden layer elements affected the output produced by the network. This study found the results to be consistent with his findings.

If sufficient amount of data is available for training, then it is possible to develop a trainable system with optimum performance using architectures currently known. The nonlinear processing capability of the neural network models add to the flexibility of such systems. The extra flexibility obtainable in neural systems is likely to lead to several practical applications for which neural network signal processors will offer better performance in a smaller, cheaper package.

CHAPTER V

RECOMMENDATIONS FOR FURTHER STUDY

The results of the study show the ability of the neural networks in signal processing applications, especially in noise filtering. In the neural network models each node does simple processing giving rise to the complex behavior for the system.

The test of the neural network based filter gave some indication of improving filtering capabilities which can be explored for other neural network architectures. Further, an additional neural network based model can be developed to classify the noise free signal of the first stage based on deviations from the normal pattern. The second stage network can then be interfaced to an expert system which can estimate the extent of the damage to the heart and alert the physician.

With some modifications, the system can be made to process real time signals. This may require creating windows, with about 20 to 40 samples. By presenting these samples to the input layer and having one unit at the output layer, a continuously filtered signal may be obtained.

The increased performance of the neural network system is achieved at the expense of increased complexity. The requirement that the input nodes be fully connected prohibits the use of digital filters when the input is large. This problem can be eliminated by the neural network filters which place no restriction on the number of units in the input layer. As progress is made in the VLSI implementation of densely connected neural network models, such neural network based filters should be able to handle complex problems.

CHAPTER VI

SUMMARY AND CONCLUSIONS

A data compression algorithm for noise filtering applications for signal processing was studied with artificial neural network systems. A three layer neural network system with one hidden layer was used. The hidden layer had fewer units than the other layers. The network was trained with the supervised signals which are the same as input signals. The backpropagation algorithm was used for training the network. The network was found to be adaptable to the environment. When the input signal changes, the network changed its weights in an effort to identify the original signal.

Neural network based techniques for noise filtering offer an interesting and potentially powerful approach. It can be extended to other signals. Depending on the type of the signal used, the processing elements in the hidden layer should use appropriate transfer function.

The performance of the network was found to be consistent with the results and findings of Klimasauskas [1989]. He showed the effect of the hidden layer elements on the output produced by the network. The ability of the

network to reconstruct the original signal was found to be determined by the hidden layer elements. The network performance improved initially with the increase in the number of elements of the hidden layer. After a certain stage the performance started to tail off gradually.

The ability of the network to learn was found to depend to a great extent on the learning rate. The learning rate should be set to a value as high as possible without causing the output error to oscillate. The optimal value of the learning rate depends on the error function in weight space. Difficult problems require usually small values of learning rate and may require many training samples.

The results of the study pointed out several new directions to investigate: the effect of multiple hidden layers on output; the effect of using other transfer functions on the hidden layer elements; and training with a large amount of decaying input noise.

This study shows an application of neural networks in identifying the wave patterns from a noisy environment. The methodology employed in the neural network filtering could also be enlarged. Other network paradigms could be incorporated into the system to improve the performance. This technique could also be extended to complex noise filtering problems and the methodology could also be extended for use with other signal processing problems.

BIBLIOGRAPHY

1. W.F. Allman, "Designing Computers the Way We Do", Technology Review, Vol.90, No.4, May/June 1987, pp 58-66.

2. J.O. Caroll, et.al., "A Neural Network Model For ECG Analysis", (Poster Abstracts), Proc. Of III International Joint Conference on Neural Nets, June 1989, pp 575.

3. M. Caudill, "The Polynomial Adaline Algorithm", Computer Language, Vol.5, No.12, Dec. 1988, pp 53-59.

4. E. Collins, et.al., "An Application of Multiple Neural Network Learning System to Emulation of Mortage Underwriting Judgements", Proc. of II International Conference of Neural Networks, July 1987, pp 459-466.

5. J.D. Cowan & D.H. Sharp, "Neural Nets and AI", Daedalus, Vol.117, No.1, Winter 1988, pp 85-122.

6. DARPA, "Neural Network Study", Oct 1987 - Feb 1988 AFCEA International Press, 1988.

7. J.W. Denton & G.R. Madley, "Impact Of Neurocomputing on Operations Research", in Impact of recent Computer Advances on OR, R. Sharda et. al., editors, Elsevier North Holland, 1989, pp 302-312.

8. S. Dutta & S. Shekar, "Bond Rating: A New Conservative Application of Neural Nets", Proc. of II International Conference of Neural Networks, July 1988, pp 443-450.

9. R.C. Eberhart, et.al., "EEG Spike Detection Using Backpropagation Networks", Proc. Of III International Joint Conference On Neural Nets, June 1989, pp 637.

10. F.A. Fazal, et.al., "Studies Of Pattern Recognition With Self-Learning Layered Neural Nets", Proc. Of III International Conference On Neural Nets, June 1989, pp 617.

11. S.I. Gallant, "Connectionist Expert Systems", Comm. Of the ACM, Vol. 31, No.2, Feb. 1988, pp 152-170.

12. G.C. Goodwin & K.S. Sin, Adaptive Filtering Prediction and Control, Prentice-Hall Inc., 1984.

13. R. Hecht-Nielsen, "Theory Of Back Propagation Neural Networks", Proc. Of III International Joint Conference On Neural Nets, June 1989, pp 593-605.

14. R. Hecht-Nielsen, "Neurocomputing: Picking the Human Brain", IEEE Spectrum, Vol.25, No.3, March 1988, pp 36-41.

15. J.J. Hopfield & D.W. Tank, "Computing with Neural Circuits: A Model", Science, Vol. 233, Aug. 1986, pp 625-634.

16. G. Hripcsak, "Problem-solving using Neural Networks", M.D. Computing: Computers in Medical Practice, Vol. 5, No. 3, 1988, pp 25-37.

17. A. Iwata, et.al., "A Digital Holter System With Dual 3 Layer Neural Network", Proc. Of III International Joint Conference on Neural Nets, June 1989, pp 69-74.

18. Leland B. Jackson, Digital Filters and Signal Processing, Second Edition, Kluwer Academic Press, 1989.

19. J. R. Johnson, Introduction To Digital Signal Processing, Prentice Hall, 1989.

20. G. Josin, "Neural Network Heuristics", Byte, Vol.12, No.10, Oct. 1987, pp 183-193.

21. J. F. Kaiser, "Systems Analysis by Digital Computer", John Wiley & Sons, 1966.

22. J. Kiwoshita & N.G. Palensky, "Computing with Neural Nets", High Technology, Vol.7, No.5, May 1987, pp 24-31.

23. C. Klimasauskas, "Neural Nets and Noise Filtering", Dr. Dobb's Journal, Vol.14, No.1, Jan 1989, pp 32-48.

24. B. Kosko, "Unsupervised Learning In Noise", Proc. Of III International Joint Conference on Neural Networks, June 1989, pp 7-17.

25. R.P. Lippman, "An Introduction to Computing with Neural Nets", IEEE ASSP Magazine, Vol.4, No.2, April 1987, pp 4-23.

26. D. Rumelhart & J.D. McClelland, Parallel Distributed Processing, Vol 1, MIT Press, 1986.

27. R. Sharda & R. Patil, "Neural Networks As Forecasting Experts: An Empirical Test", International Joint Conference On Neural Networks, Jan. 1990.

28. J.L. Solka, et.al., "Signal Processing With Neural Nets: Part I Signal In Noise", Proc. Of III International Conference On Neural Nets, June 1989, pp 609.

29. N. Sonehara,et.al., "Image Data Compression Using Neural Networks Model", Proc. III International Conference On Neural Networks, June 1989, pp 35-41.

30. J. Stanley, "Introduction to Neural Networks", California Scientific Software, 1989.

31. D.F. Stubbs, "3 - Applications of Neurocomputing in Biomedical Research", Proc. Of III International Conference On Neural Nets, June 1989, pp 609.

32. P.J. Werbos, "Back Propagation & Neuro Control: A Review and Prospectus", Proc Of III International Conference On Neural Networks, June 1989, pp 209-216.

33. B. Widrow, et. al., "Adaptive Noise Cancelling: Principles and Applications", Proceedings of the IEEE, Vol. 63, No. 12, Dec 1975.

34. B. Widrow and S. D. Stearns, Adaptive Signal Processing, Prentice-Hall Signal Processing Series, 1985.

35. B. Widrow and R. Winter, "Neural Networks for Adaptive Filtering & Adaptive Pattern Recognition", IEEE Computer, Vol. 21, No. 3, 1988, pp 34-41.

36. M.C. Yeates, "An Architecture with Neural Nets Characteristics for Least Square Problem", 1989. (To Appear in IEEE ASSP magazine)

37. _____, SAIC's ANSIM User's Manual, Science Applications International Corporation, 1988.

APPENDICES

APPENDIX A

DERIVATION OF THE DELTA RULE

DERIVATION OF THE DELTA RULE

The Delta rule can be derived in many ways. The following derivation applies to linear units that minimize the squares of the differences between the actual output produced by the output units and the desired output values for all pairs of input/output patterns. This can be shown showing the derivative of the error measure with respect to each weight is proportional to the weight change as dictated by the delta rule, the constant of proportionality being negative. This actually corresponds to performing steepest descent on a surface in weight space whose height at any point in error surface equals error measure.

The error, E_p, produced by the pattern set, p, is

$$E_p = 1/2 \sum(t_{pJ} - O_{pJ})^2 \qquad (6)$$

where t_{pJ} is the desired input for the jth component of the output pattern.

O_{pJ} is the actual output produced by unit j for pattern p.

The overall error is $E = \sum E_p$.

When the units are linear,

$$\partial E_p / \partial W_{iJ} = \delta_{pJ}\ i_{pJ} \qquad (7)$$

By applying the chain rule, we have,

$$\partial E_p/\partial W_{ji} = \partial E_p/\partial O_{pj} \cdot \partial O_{pj}/\partial W_{ji} \quad (8)$$

$\partial E_p/\partial O_{pj}$ indicates the error changes with respect to the output of the jth unit.

$\partial O_{pj}/\partial W_{ji}$ shows how much of changing weights changes that output.

From equation (9) we get,

$$\partial E_p/\partial O_{pj} = 1/2 \cdot 2 (t_{pj} - O_{pj}) (-1)$$

$$= -(t_{pj} - O_{pj})$$

$$= -\delta_{pj} \quad (9)$$

For a set of linear units,

$$O_{pj} = \sum W_{ji} \cdot i_{pj}$$

Taking derivatives with respect to W_{ij},

$$\partial O_{pj}/\partial W_{ji} = i_{pj} \quad (10)$$

Substituting (12) and (13) in equation (11),

$$\partial E_p/\partial W_{ji} = -\delta_{pj} \cdot i_{pj}$$

$$-\partial E_p/\partial W_{ij} = \delta_{pj} i_{pj}$$

Combining for all input/output patterns

$$\partial E/W_{ji} = \sum \partial E_p/\partial W_{ji} \quad (11)$$

i.e. the net change in the weight vector, W_{ji}, for each cycle of training is proportional to this derivative. Hence, the delta rule implements a gradient descent in error, E.

APPENDIX B

BACKPROPAGATION TRAINING ALGORITHM

BACKPROPAGATION TRAINING ALGORITHM

The backpropagation training algorithm minimizes the mean square error between the actual output produced by the multilayered neural networks and the target output. It is an interactive gradient descent algorithm. The algorithm is based on sigmoid activation function and is given by Lippman [LIPPMAN, 1987].

Step 1: Initialize weights and offsets

The weights and node offsets are initialized to a random value within specified limits.

Step 2: Apply input and output vector patterns

The training patterns contain a set of vectors, an input pattern, I, and desired output pattern, T.

$$I = [i_0 \ldots i_{N-1}]$$

$$T = [t_0 \ldots t_{M-1}]$$

where N is number of units in the hidden layer, and M is the number of units in the output layer.

The training should be presented cyclically till the weights subsidize.

Step 3: Obtain actual outputs

Determine the output, Y, based on the sigmoid nonlinearity of the processing unit.

Step 4: Adapt weights

By using recursive algorithms, working back from the output layer to the first hidden layer, adjust weights according to

$$W_{ij}(t+1) = \eta(\delta_j X_j) + W_{ij}(t) \quad (12)$$

If the node, j, is the output node, then

$$\delta_j = Y_i(1 - Y_i).\ (t_i - Y_i) \quad (13)$$

If the node, j, is an internal hidden node, then,

$$\delta_j = X'_j\ (1 - X'_j)\ \sum \delta_k . W_{jk}, \quad (14)$$

where X'_j is either output of the node i or is an input k is over all nodes in the layers above node j.

The convergence can be made faster by providing a momentum and smoothing weight change by

$$W_{ij}(t+1) = W_{ij}(t) + \eta \delta_j X'_j + \alpha(W_{ij}(t) - W_{ij}(t-1)),$$

where $0 < \alpha < 1$.

Step 5: Repeat again starting at step 2.

APPENDIX C

THE ADAPTIVE LINEAR COMBINER

THE ADAPTIVE LINEAR COMBINER

The adaptive linear combiner is fundamental to adaptive signal processing systems and filters. It is the single most important element in learning systems and adaptive processes. A general form of adaptive linear combiner is shown in Figure 5.

The adaptive linear combiner consists of input signal vector, $[X_0, X_1, X_3, \ldots X_L]$, a corresponding set of adjustable weights, $[W_0, W_1, \ldots WL]$, a summing unit, and a single output signal Y. A procedure for adapting weights is called an adaptation procedure. The combiner is called linear because for a fixed weight setting its output is a linear combination of the input components. When the weights are in the process of being adjusted, they, too are a function of the input components and the output of the combiner is no longer a linear function of the input. The operation becomes non-linear.

There are two different types of input vectors, multiple and single input. Multiple input may be considered to be simultaneous inputs from different sources. Single input may be considered to be sequential samples from the same signal source. The rest of the discussion deals with the single input, as multiple input is almost similar to the

single input combiner.

Let X_k represent the single input signal.

$$X_k = [X_k, X_{k-1}, \ldots X_{K-L}]^T \quad (15)$$

In this notation, T stands for transpose, so X_k is actually a column vector in both cases. The subscript K is used as a time index. In the case of single input, the adaptive processor can be implemented with an adaptive linear combiner and unit delay elements as shown in Figure 4. This structure is called an adaptive traversal filter.

For the input signal represented in (18) the input-output relation is as follows:

$$Y_k = \sum W_{1k} \cdot X_{k-1} \quad (16)$$

where $W_k = [W_{0k}, W_k, \ldots W_{LK}]^T$ is the weight vector.

Using Vector representation, equation (19) can be rewritten as

$$Y_k = X_k \cdot W_k^T \quad (17)$$

In the adaptation process with the performance feedback, the weight vector of the linear combiner is adjusted to cause the output, Y_k, to agree as closely possible with the desired response signal. This is done by comparing the output with the desired response to obtain an error signal and then adjusting the weight vectors to minimize the signal. The method of deriving the error signals is to subtract the output signal, Y_k, from the target signal d_k, to produce an error signal E_k. Mathematically this can be

given as

$$E_k = d_k - Y_k. \tag{18}$$

APPENDIX D

PRINCIPLES OF FIR FILTER

PRINCIPLES OF FIR FILTER

In appendix A, the adaptive linear combiner and its properties were described without recourse to the usual transform or frequency domain analysis. The analysis of adaptive signal processing involves Z-transforms, as Z-transforms transform directly relates to the frequency response of the system.

Most signal processing systems involve the use of sample sets include input, desired and error signals as well as the set of weight vectors. The Z-transforms of any such sequence is defined as follow:

Data sample set: $[X_k] = [\ldots X_{-1}, X_0, X_1, X_2, \ldots]$ (19)

Z-transform: $X(Z) = \sum X_k . Z^{-k}$ (20)

Where Z is a continuous complex variable,

X(Z) is a two sided Z-transform because negative as well as positive values of index k is involved.

For describing the design principles, the following terminology are used.

Transfer function : The transform of the output of the system divided by the transform of the input. The transform used here is the Z-transform.

$h_d(n)$ is the impulse response of the system.

$H_d(n)$ is the desired frequency response of the system.

$H(w)$ is the Z-transform of $h(n)$ evaluated on the unit circle of pole-zero plot, ie.,

$H_d(w) = \sum h_d(n)\ e^{-jwn}$.

Window - Function Technique

The most obvious way to design an FIR filter is to truncate the ideal response $h_d(n)$ outside the interval $0 <= n <= M$ to produce $h(n)$, ie.,

$$h(n) = h_d(n) \quad 0 <= n <= M \qquad (21)$$
$$= 0 \quad \text{otherwise}$$

This results in an expression, which is equivalent to the above expression is

$$h(n) = \omega_R(n).\ h_d(n) \qquad (22)$$

where $\omega_R(n)$ is the rectangular window, defined as

$\omega_R(n) = 1\ 0 <= n <= M$

$= 0$ otherwise

The relation between the two impulse-response sequences corresponding to equation (25) is given by

$h_S(n) = \omega_S(n).\ h_d(n)$

where the subscript, S, indicates the symmetry of the window and it distinguishes from the casual window $\omega_R(n)$. There are other windowing techniques like Hann window, Hamming window, Blackman window and Kaiser window. The analysis of these windows is beyond the scope of this study.

Frequency Sampling Technique

Another method that is used in the design and analysis of FIR filters is the frequency sampling. In this method a set of samples is determined from a desired frequency response and is identified as discrete Fourier Transform (DFT) coefficients. The filter coefficients are then determined as the inverse discrete Fourier-Transform (IDFT) of this set of samples.

Optimal FIR Filters

The design and analysis for optimal filters is based on a minimax or Chebyshev type approximation. This technique involves the determination of a weighted error function based on the desired response and the general form of the response function. The coefficients in the response function are then determined to minimize the maximum error that occurs.

APPENDIX E

GLOSSARY

GLOSSARY

Adaptive coefficients

The values computed during previous training are stored in a local memory which are used for subsequent modifications.

Adaptive filter

The pattern of activity in neurons are transferred from one level to that in another through the set of interconnections that are formed between neurons in one level and those at another level. This interconnection is called a filter.

Backpropagation

The information processing is the approximation of a mapping or function f: $A \subset R^n \rightarrow R^m$, from a bounded subset A of n-dimensional Euclidean space to a bounded subset f[A] of m-dimensional Euclidean space, by means of training samples $(X_1,Y_1),(X_2,Y_2)\ldots(X_k,Y_k),\ldots$ of the mapping's action where $Y_k = f(X_k)$. It is assumed that such samples of a mapping f, are generated by selecting X_k vectors randomly from A in accordance with fixed probability density function P(x). [R. HECHT-NIELSEN, 1989].

Connection

The pathway formed between the neurons for signal transmission.

Feature detector

The response of the neuron when a particular pattern of stimulus is present. For example the feature might correspond to a pixel in image processing or a particular acoustic frequency component in acoustic signals.

Graceful Degradation

In neural networks, there is no single point at which the performance breaks down. The network performance gradually deteriorates as more and more processing elements are destroyed.

Hidden layer

The layer of neurons formed between the input and output layers. They are called hidden because they derive the input from other layers and feed their output to other layers. A neural network model may have one or more hidden layers.

Input layer

The layer of neurons to which the input pattern is applied for processing.

Input pattern

The input stimulus presented to the neural network for processing.

Layers

The neurons in the neural network are arranged into layers. The neurons in a layer have similar transfer function.

Learning law

It is the equation which determines that all or some of the weights in the neuron's local memory be modified with response to the input signal and transfer function of that neuron.

Neural network

A neural network is a system in which many neurons process the information in a parallel manner. The overall function or response of the system is determined by weights present in the connection, and the processing done at the computing elements or neurons.

Neuron

The fundamental unit in the neural network. It is a nonlinear computational unit named after its counterpart in brain.

Output layer

The layer of neurons which presents the output response of the neural network.

Self-supervised Training

Self-supervision is used in automata which require internal error feedback to perform some specific task. The learning is autonomous and no external correct input is specified.

Supervised Learning

The learning in the neural networks where the correct or expected response is provided by an external teacher at the time of training.

Transfer function

The response of the neuron depends on the mathematical formula and is a function of the most recent input signals and the adaptive weights stored in memory.

Unsupervised learning

The learning in the neural networks where no external teacher provides the correct response.

Weights

The strength of the interconnection between neurons are determined by a variable, called weights. The weights determine the intensity of connection, which depends on network architecture and the information it has learned.

VITA

Ranganathan Ramkumar

Candidate for the Degree of

Master Of Science

Thesis: A TEST OF NOISE FILTERING CAPABILITIES OF BACKPROPAGATION NEURAL NETWORKS

Major Field: Computing and Information Sciences

Biographical:

Personal Data: Born in Udumalpet, Tamil Nadu, India, July 5, 1965, the son of S. Ranganathan and Vanithamani.

Education: Graduated from Mani Higher Secondary School, India, 1982; received Bachelor of Engineering with a major in Electronics and Communication Engineering from the P. S. G. College of Technology in July, 1986; completed requirements for the Master of Science degree at Oklahoma State University in May, 1990.

Professional Experience: Research Assistant, Oklahoma Resources Integrated General Information System (ORIGINS), Oklahoma State University, August 1988 to October 1989; Asst. system administrator, State 4-H Programs, Oklahoma State University, July 1987 to June 1988; Trainee Engineer, Festo-Elgi Pvt. Limited, India, April 1986 to December 1986.