Performance Evaluation of Generalized ADALINE Neural Configuration with Variable Learning-rate Parameter

Thesis submitted in the partial fulfillment of requirement for the award of degree of

MASTER OF ENGINEERING

In

ELECTRONICS & COMMUNICATION ENGINEERING

Submitted by

Savita

Roll No. 801161026

Under the Guidance of

Dr. Amit Kumar Kohli

Assistant Professor, ECED, TU

Electronics and Communication Engineering Department

Thapar University, Patiala – 147004 (PUNJAB)

June- 2013

CERTIFICATE

I, **Savita** hereby certify that the work, which is being presented in this thesis entitled "**Performance Evaluation of Generalized ADALINE Neural Configuration with Variable Learning-rate Parameter**" by me in partial fulfillment of the requirements for the award of degree of Master of Engineering in Electronics and Communication Engineering at Thapar University, Patiala, is an authentic record of my own work carried out under the guidance of **Dr. Amit Kumar Kohli,** Assistant Professor, **Electronics and Communication Engineering Department**, Thapar University, Patiala.

The matter presented in this thesis has not been submitted in any other university or institute for the award of the degree of Master of Engineering.

Savita

Savita

Date 28/6/2013

This is certified that the above statement made by the candidate is correct to the best of my knowledge.

Amit Kohli

Dr. Amit Kumar Kohli
Assistant Professor, ECED
Thapar University, Patiala
Date 28/06/2013

Countersigned by:

Dr. R. Khanna
Professor & Head, ECED,
Thapar University, Patiala
Date 2/7/13

Dr. S.K. Mohapatra
Dean of Academic Affairs
Thapar University, Patiala
Date 3/7/13

ACKNOWLEDGEMENT

No volume of words is enough to express my gratitude towards my guide, **Dr. Amit Kumar Kohli**, Assistant Professor, Electronics and Communication Engineering Department, Thapar University, who has been very concerned and has aided for all the material essential for the preparation of this thesis report. He has helped me to explore this vast topic in an organized manner and provided me with all the ideas on how to work towards a research-oriented venture.

I am also thankful to **Dr. Rajesh Khanna**, Head of Department, ECED, **Dr. Kulbir Singh**, P.G. Coordinator and **Dr. Sanjay Sharma**, Professor, Electronics and Communication Engineering Department, Thapar University, for the motivation and inspiration that triggered me for the thesis work.

I would also like to thank the staff members and my colleagues who were always there in the need of the hour and provided with all the help and facilities, which I required, for the completion of my thesis.

Most importantly, I would like to thank my parents and the Almighty for showing me the right direction out of the blue, to help me stay calm in the oddest of the times and keep moving even at times when there was no hope.

Savita

Roll No. **801161026**

ABSTRACT

Artificial Neural Network (ANN) is a mathematical model that is inspired by the structured and/or functional aspects of biological neural networks. A neural network consists of an interconnected group of artificial neurons, and it processes information using a connectionist approach to computation. Recent advances in the software and hardware technologies of neural networks have motivated new studies in the architecture and applications of these networks. Neural networks have potentially powerful characteristics, which can be utilized in the development of our research goal, namely, a true autonomous machine. Machine learning is a major step in this development. The immense computational power of modern digital machines has increased the feasibility of implementing closed-loop control systems in many applications. The main focus of this thesis work is completed on the performance evaluation *ADALINE neural configuration* for system identification of dynamical systems, noise cancellation and adaptive prediction.

It is well known that ADALINE is slow in the convergence which is not appropriate for the online applications and identification of the time varying systems. Two techniques are proposed to speed up the convergence of learning, thus increase the tracking capability of the time varying system parameters. One idea is to introduce a momentum term to the weight adjustment during convergence period. The other technique is to train the generalized ADALINE network multiple epochs with data from a sliding window of the system's input output data.

We present an online identification method based on a generalized Adaptive Linear Element (ADALINE) neural network, called GADALINE, for linear time varying systems which can be described by a discrete time model. The fine tuned GADLINE is quite suitable for online system identification and real time adaptive control applications due to its low computational demand.

GADALINE neural configuration with a variable learning-rate parameter is introduced. This parameter increases or decreases as the mean-square error increases or decreases, allowing the adaptive filter to track changes in the system as well as to produce small steady state error.

Therefore, we have made an effort to do analysis of different techniques with variable learning- rate parameter implemented on GADALINE neural configuration. This work discusses the motivations behind the development of ANNs and describes the basic biological neuron and the artificial computational model. The thesis concludes a comparison of the results obtained for three different techniques. These three algorithms use variable learning-rate parameter, which are already proved to have better performance over algorithm with fixed learning-rate parameter.

In the first algorithm learning-rate parameter adjustment is controlled by the square of the prediction error. The second algorithm adjusts the learning-rate parameter according to a gradient descent algorithm, designed to reduce the squared estimation error during each iteration. In the third algorithm the learning-rate parameter of the algorithm is adjusted according to the square of a time-averaging estimate of the autocorrelation of e(n) and e(n-1). Simulation results show that third algorithm gives the best result over other algorithms in a stationary environment for the same excess MSE under similar environments.

Keywords*: ADALINE, learning-rate parameter, gradient descent algorithm, MSE, GADALINE.*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

ANN	Artificial Neural Network
VLSI	Very Large Scale Integration
SISO	Single Input Single Output
SLNN	Single Layer Neural Network
LTI	Linear Time Invariant
MCP	McCulloch and Pitts
ADALINE	Adaptive Linear Element
GADALINE	Generalized Adaptive Linear Element
TDL	Tapped Delay Line
LS	Least Squares
LMS	Least Mean Squares
NLMS	Normalized Least Mean Squares
MSE	Mean Square Error
LMSE	Least Mean Square Error
RLS	Recursive Least Squares
ALC	Adaptive Linear Combiner
GD	Gradient Descent
SNR	Signal to Noise Ratio
ANC	Adaptive Noise Canceller

CHAPTER 1

INTRODUCTION

1.1 Motivation

Human brain is the most extraordinary and complex creation in the universe. It has made human beings stand apart from the animal kingdom. The human brain, being the most intelligent device on the earth, drives us being the ever-progressive species on the planet. The advantage of human brain is its massive parallelism, the highly parallel computing structure. The human brain is a collection of approximately 10^{11} computing elements called neurons (shown in figure 1). Neurons are living cells with axons (single long fiber) and dendrites (tree like networks of nerve fibers) that form interconnections through electro-chemical synapses, with a density of approximately 10^4 synapses per neuron. Signals are transmitted through the cell body (soma), from the dendrite to the axon as an electrical impulse, by raising or lowering the electric potential inside the body of the receiving cell. If the potential reaches a threshold, a pulse is sent through axon and the cell is sad to have 'fired'.

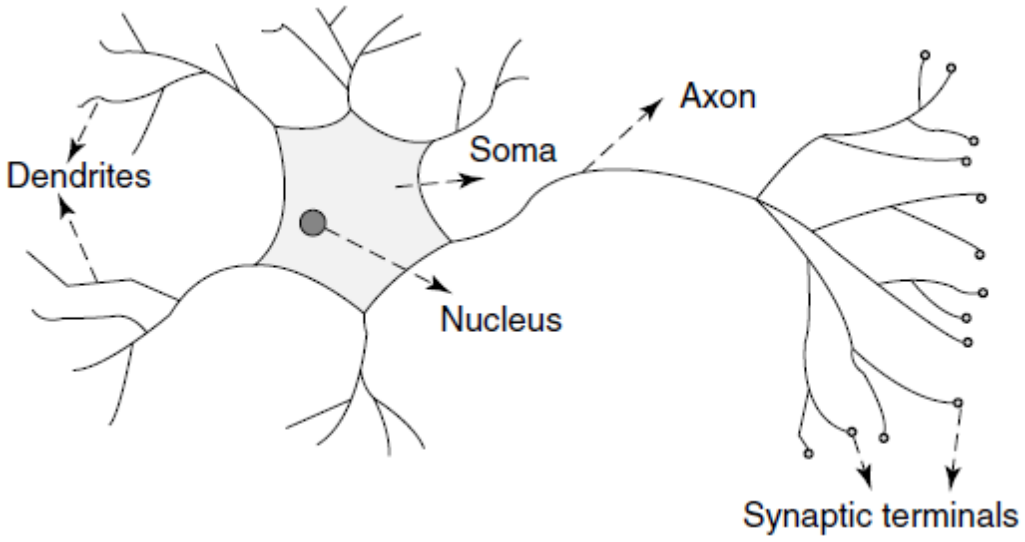

Figure 1 *Biological neuron [1]*

Man always tried to make machines that could do intelligent job processing, and took decisions on their own. The result was a Computer. Even though it could perform millions of calculations

every second, display incredible graphics and 3-dimentional animations, play audio and video but it made the same mistake every time. Practice could not make it perfect. So the question for making more intelligent device continued. Then the idea of initiating human brain stuck the designers who started their researches, giving rise to ***Artificial Neural Networks***.
So, borrowing from biology, researchers are exploring neural networks—a new, non algorithmic approach to information processing.

A Neural Network is a powerful data-modeling tool that is able to capture and represent complex input/output relationships. The motivation for the development of neural network technology stemmed from the desire to develop an artificial system that could perform "intelligent" tasks similar to those performed by the human brain. Neural networks resemble the human brain in the following two ways:

- A neural network acquires knowledge through learning.
- A neural network's knowledge is stored within inter-neuron connection strengths known as synaptic weights [2].

Synthetic networks that emulate the biological neural networks found in living organisms are called Artificial Neural Networks. Artificial neural networks have undoubtedly been biologically inspired, but the close correspondence between them and real neural systems is still rather weak. A neural network consists of an interconnected group of artificial neurons, and it processes information using a connectionist approach to computation. In most cases an ANN is an adaptive system that changes its structure based on external or internal information that flows through the network during the learning phase. Modern neural networks are non-linear statistical data modeling tools. They are usually used to model complex relationships between inputs and outputs or to find patterns in data.

The above mentioned properties of an ANN serve as a primary motivation for their on-chip implementation. This work comprehensively summarizes the efforts towards the implementation of individual ANN modules.

1.2 Artificial Neural Network

Artificial Neural Network (ANN) is a mathematical model that is inspired by the structured and/or functional aspects of biological neural networks. A neural network consists of an interconnected group of artificial neurons, and it processes information using a connectionist approach to computation. The block diagram of Figure 2 shows the mathematical model of a neuron, which forms the basis for designing ANNs. Here we identify three basic elements of the neural model:

1. A set of ***Synapses*** or connecting Links, each of which is characterized by a ***Weight*** or ***Strength*** of its own. Specifically, a signal $\mathbf{X_i}$ at the input of synapse ***I*** connected to neuron ***k*** is multiplied by the synaptic weight $\mathbf{W_{ki}}$.
2. An ***Adder*** for summing the input signals, weighted by the respective synapses of the neuron.
3. An ***Activation Function*** for limiting the amplitude of the output of a neuron. The activation function is also referred to as a ***Squashing Function*** in that it squashes (limits) the permissible amplitude range of the output signal to some finite value.

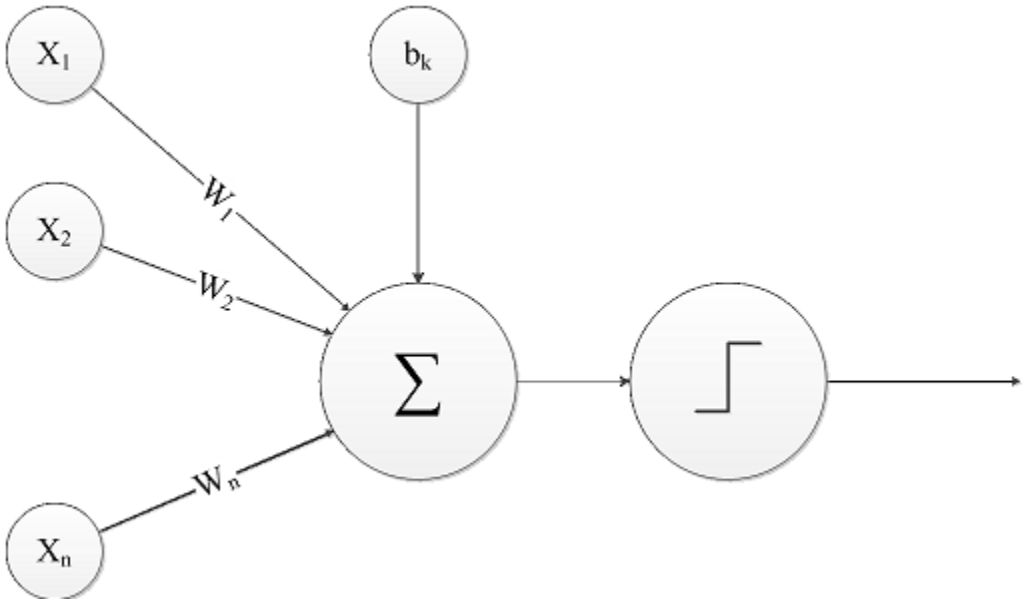

Figure 2 *Mathematical Model of Neuron [2]*

The neural model of Figure 2 also includes an externally applied Bias, denoted by $\boldsymbol{B_k}$. The bias $\boldsymbol{B_k}$ has the effect of increasing or lowering the net input of the activation function, depending on

whether it is positive or negative, respectively. In mathematical terms, we may describe a neuron *k* by writing the following pair of equations:

$$u_k = \sum w_{ki} \times x_i + b_k \quad (1.1)$$

and

$$y_k = \mathrm{f}(u_k) \quad (1.2)$$

where X_i are the input signals; W_{ki} are the synaptic weights of neuron *k*; B_k is the bias; U_k is the adder output, f (·) is the *activation function*; and Y_k is the output of the neuron. The use of bias B_k has the effect of applying an affine transformation to the output U_k of the linear combiner in the model of Figure 2.

The immense computational power of modern digital machines has increased the feasibility of implementing closed-loop control systems in many applications. However, in some cases, as the number of sensor inputs increase the on-line computational requirements approach the intensity of pattern recognition. It has been shown that even the fastest modern computer cannot compare to the brain of an infant in such performances as image processing and pattern recognition. As an alternative form of information processing, neuron computing is becoming an established discipline. Although artificial neural network [3] is only a metaphorical resemblance to biological neural systems, it offers the advantage of parallel distributed processing of vast amount of information and performance improvement through learning. A neural network is a massively parallel distributed processor made up of simple processing units, called neurons, which has the natural propensity for storing experiential knowledge and making it available for use [2].

1.3 Applications of Neural Networks

In order to have an integrated understanding on neural networks, we adopt the next perspective, called top-down, from application, algorithm to architecture. The approach is application-motivated, theoretically based, and implementation oriented. The main applications are for signal processing and pattern recognition. The algorithmic treatment represents a combination of

mathematical theory and heuristic justification for neural models. The ultimate objective is the implementation of digital neuro computers, embracing technologies of VLSI, adaptive, digital and parallel processing.

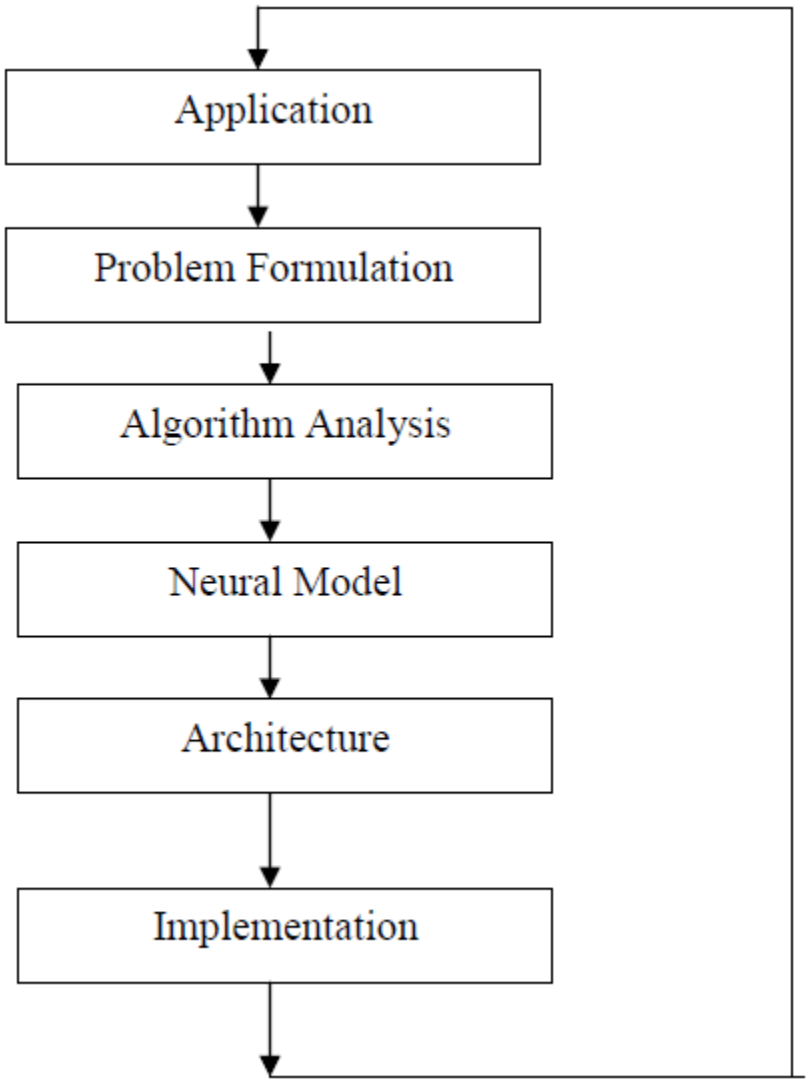

Figure 3 *Flowchart for Implementation of Applications*

1.3.1 Neural Networks in Practice

Neural networks have broad applicability to real world business problems. In fact, they have already been successfully applied in many industries. Since neural networks are best at identifying patterns or trends in data, they are well suited for prediction or forecasting needs including:

- Sales Forecasting
- Industrial Process Control
- Customer Research
- Data Validation
- Risk Management
- Target Marketing

But to give you some more specific examples; ANN are also used in the following specific paradigms: recognition of speakers in communications; diagnosis of hepatitis; recovery of telecommunications from faulty software; interpretation of multimeaning Chinese words; undersea mine detection; texture analysis; three-dimensional object recognition; hand-written word recognition; and facial recognition.

1.3.2 Neural Networks in Medicine

Artificial Neural Networks (ANN) is currently a 'hot' research area in medicine and it is believed that they will receive extensive application to biomedical systems in the next few years. At the moment, the research is mostly on modeling parts of the human body and recognizing diseases from various scans (e.g. cardiograms, CAT scans, ultrasonic scans, etc.) [4,5].

Neural networks are ideal in recognizing diseases using scans since there is no need to provide a specific algorithm on how to identify the disease. Neural networks learn by example so the details of how to recognize the disease are not needed. What is needed is a set of examples that are representative of all the variations of the disease. The quantity of examples is not as important as the 'quantity'. The examples need to be selected very carefully if the system is to perform reliably and efficiently.

1.3.2.1 Modeling and Diagnosing the Cardiovascular System

Neural Networks are used experimentally to model the human cardiovascular system. Diagnosis can be achieved by building a model of the cardiovascular system of an individual and

comparing it with the real time physiological measurements taken from the patient. If this routine is carried out regularly, potential harmful medical condition scan be detected at an early stage and thus make the process of combating the disease much easier.

A model of an individual's cardiovascular system must mimic the relationship among physiological variables (i.e., heart rate, systolic and diastolic blood pressures, and breathing rate) at different physical activity levels. If a model is adapted to an individual, then it becomes a model of the physical condition of that individual. The simulator will have to be able to adapt the features of any individual without the supervision of an expert. Here, it needs a neural network.

Another reason that justifies the use of ANN technology is the ability of Ann's to provide sensor fusion which is the combining of values from several different sensors. Sensor fusion enables the Ann's to learn complex relationships among the individual sensor values, which would otherwise be lost if the values were individually analyzed. In medical modeling and diagnosis, this implies that even though each sensor in a set may be sensitive only to a specific physiological variable, Ann's are capable of detecting complex medical conditions by fusing the data from the individual biomedical sensors.

1.3.2.2 Electronic Noses

Ann's are used experimentally to implement electronic noses. Electronic noses have several potential applications in telemedicine. Telemedicine is the practice of medicine over long distances via a communication link. The electronic nose would identify odors in the remote surgical environment. These identified odors would then be electronically transmitted to another site where a door generation system would recreate them. Because the sense of smell can be an important sense to the surgeon, telesmell would enhance telepresent surgery.

1.3.2.3 Instant Physician

An application developed in the mid-1980s called the "instant physician" trained an auto associative memory neural network to store a large number of medical records, each of which includes information on symptoms, diagnosis, and treatment for a particular case. After training, the net can be presented with input consisting of a set of symptoms; it will then find the full stored pattern that represents the "best" diagnosis and treatment.

1.3.3 Neural Networks in Business

Business is a diverted field with several general areas of specializations such as accounting or financial analysis. There is some potential for using neural networks for business purposes, including resource allocation and scheduling. There is also a strong potential for using neural networks for database mining that is, searching for patterns implicit within the explicitly stored information in databases. Most of the funded work in this area is classified as proprietary. Thus, it is not possible to report on the full extent of the work going on. Most work is applying neural networks, such as the Hopfield-Tank network for optimization and scheduling [6].

1.3.4 System Identification using Neural Network

In general, system identification is to determine the model structure and parameters for a dynamic system based on the measurable input and output data of the system [7]. Model structure in its simplest form for a single input single output (SISO) linear system is determined by the system order. Once the order is determined, identification reduces to a parameter estimation problem which can be well solved by traditional identification methods, such as least squares method, maximum likelihood method and instrumental variable method [8]. Most neural networks used for system identification are Hopfield [9,10], Radial Basis Function [11], Support Vector [12], Self-Organizing Map [13].

A technique for programming of Hopfield network for the purpose of system identification was developed. Spontaneous energy minimization by Hopfield network is used to minimize least-mean-square of error rates of estimates of state variables. This method estimates parameters of mathematical model of a system or process, linearly independency is also ensured [14]. Linear identification is also useful for adaptive control [15,16,17] .

When the focus is on the applications of neural networks for nonlinear system modeling and identification[18,19]: Hopfield network for identifying a linear time invariant dynamical system [20], for some problems, globally optimal solution is not guaranteed; the network computes locally optimum solutions [21,22]. In such cases it might not be advisable to use this network, as it gives quite in accurate results and is computationally complex [23,24].

Single layer neural network (SLNN) for identifying the parameters of a linear system formulated least mean square error (LMSE) as a performance measure for the network. This network is simpler in structure and, therefore, less expensive in implementation. The circuit is suitable for on-line computation because of the parallel nature of its architecture and possibility of the use of analog circuit elements.

Generalized Adaptive Linear Element (ADALINE) neural network, called GADALINE, for linear time varying systems which can be described by a discrete time model. In such systems, the current system output is dependent on past outputs and on both the current and past inputs. So the GADALINE needs to have output feedback and to remember past input/output data. The proposed neural network employed a tapped delay line (TDL) for both input and output of the system to remember the past input/output data. GADALINE is then a linear recurrent type of neural network. The adaptive learning is generalized by adding a momentum term to the GADALINE weight adjustment. This momentum term is turned on only during convergence period and the learning curve is therefore smoothed by turning off the momentum once the learning error is within a given small number – epsilon. By using each set of samples for several epochs obtained from a sliding window of the system's input output data further generalizes the online training.

GADALINE provides a much faster convergence speed and better tracking of time varying parameters. Due to the simple structure of the ADALINE, the learning algorithm has an exceptionally low computational complexity and it makes this method suitable for online system identification and real time adaptive control applications.

1.3.5 Noise Cancellation using Neural Network

Noise cancellation technology is one of the major problems of signal processing. It is one of the most common practical applications of adaptive filters. Here, we use a simple neural network called ADALINE as adaptive filter. ADALINE was developed by Widrow and Hoff, who gave the name ADALINE to adaptive linear elements [24]. There are many noise cancellation applications which require utilization of adaptive filters. The LMS algorithm which is one of the most efficient criteria for determining the values of the adaptive noise cancellation coefficients are very important in communication systems, but the LMS adaptive noise cancellation suffers response degrades and slow convergence rate. Therefore, we present an adaptive noise canceller algorithm based neural network. An adaptive noise canceller adaptively filters a noise reference input to maximally match and subtract out noise or interference from the primary (signal plus noise) input. Adaptive filters have the ability to adjust their own parameters automatically and their design requires little or no a priori knowledge of signal or noise characteristics [25,26]. We can use ADALINE neural network as adaptive filter to cancel engine noise in a car. The proposed algorithm can also be applied to noise cancelling problem of long distance communication channel and also, in the development of a noise cancellation headphone.

1.3.6 Adaptive Prediction using Neural Network

Adaptive filtering is applied to prediction because of its ability to track and converge upon the stochastic characteristics of a signal [24]. In nonstationary environments, adaptive algorithms provide a frequent coefficient update that follows changes in signal statistics. This technique in adaptive filtering is efficient in terms of reduction of computational complexity and increased convergence speed. Prediction and adaptive filtering have been successfully employed in many applications such as speech coding, time series modeling, and spectrum estimation. The normalized least mean squares (NLMS) and the recursive least squares (RLS) adaptation

algorithms are two popular algorithms for updating linear adaptive filter weights [24]. The NLMS algorithm is a stochastic gradient type algorithm that relatively low complexity requiring O(3P) multiplications where P is the adaptive filter order; however, the convergence speed of the NLMS is slowed by input signals with wide eigen value spread. The RLS algorithm is more complex than NLMS requiring $O(P^2 + 5P)$ multiplications, but its convergence is independent of eigen value spread [27].

1.4 Problem Statement

This thesis report presents the following work

1. Firstly, Artificial Neural Network and Neural Signal Processing are discussed.
2. Next, the focus will be on Generalized ADALINE model.
3. Subsequently, three variable learning-rate parameter algorithms are investigated.
4. Further, simulations are performed to verify the characteristics of variable learning-rate parameter algorithms under similar conditions.

1.5 Organization of Report

This report is organized in six chapters

- Chapter I summarizes the basic problem statement of the research work and gives the overview of ANN and its applications, which we have considered in this thesis work: System Identification, Noise cancellation, Adaptive prediction.
- Chapter II, includes literature survey based on papers of current and previous years.
- Chapter III introduces ADALINE and GADALINE neural configurations.
- Chapter IV discusses the three neural adaption learning algorithms and their incorporation in different applications in brief.
- Chapter V demonstrates simulation results related to the above methods and their comparison for each application
- Chapter VI manifests the concluding remarks and future scope.

CHAPTER 2

LITERATURE SURVEY

Neural Network

Neural networks in nature are not designed but evolved, and they should learn their structures through the interaction with their environment. The ability to learn is the most important property of the living systems. Evolution and learning are the two most fundamental adaptation processes and their relationship is very complex. Studying the evolution and development processes of biological systems can reveal how the structures are formed through interactions with the environment of the living systems in nature [28]. These structural adaptation mechanisms in biological systems can suggest ways of building adaptable structure so that finally the network grows to a configuration using which suitable class of problems can be characterized by the training patterns. The initial step is to explore the aspects of this relationship by defining the process of evolution as the process of learning procedure to be adjusted [29]. This paper showed how an adaptive neural network model with reflection can implement adaptive processes. Here we understand the concept of an adaptive function, and learning mechanisms were understood in terms of their specific adaptive functions. A neural network module can be modeled to adjust its internal state to evolving training environments by modifying its adaptive function. The learning process itself is the object of evolution. Current neural network models allow the networks to adjust their behavior by changing the interconnections weights, associating neurons to each other.

ADALINE and GADALINE

A neural network is a massively parallel distributed processor made up of simple processing units, called neurons, which has the natural propensity for storing experiential knowledge and making it available for use [2]. ADALINE was developed by Widrow and Hoff, who gave the name ADALINE to adaptive linear elements [1]. ADALINE is adaptively trained to minimize

mean-square error $E[e^2(k)]$. The minimization can be achieved by adjusting the filter coefficients in adaptive filters or weights in the neural network. Due to the simple structure of the ADALINE, the learning algorithm has an exceptionally low computational complexity and it makes this method suitable for real time adaptive control applications. It is well known ADALINE is slow in convergence which is not appropriate for online application of time varying system. We need to modify both the structure and the learning algorithm of the ADALINE. So, modified structure GADALINE is being used to speed up convergence of learning and thus increases the capability of tracking time varying parameters for which two techniques are proposed, *i.e.* i) a momentum term added to the weight adjustment and ii) training on a sliding window over data set [30]. The paper showed GADALINE is a simple linear neural network which can be trained to adaptively model a linear or non-linear system. Adaptive training (also called online/incremental training, as opposed to batch training) can be used so that any change in parameters of the system also changes the weights of the neural network. Thus, at any instant of time, GADALINE can track changes in the parameters of a linear time varying system.

Neural Networks for System Identification

In general, system identification is to determine the model structure and parameters for a dynamic system based on the measurable input and output data of the system. Model structure in its simplest form for a single input single output (SISO) linear system is determined by the system order. Once the order is determined, identification reduces to a parameter estimation problem which can be well solved by traditional identification methods, such as least squares method, maximum likelihood method and instrumental variable method.

Two approaches for learning and identification of dynamical systems are presented. Hopfield and Tank [31,32] demonstrated that some classes of optimization problems can be programmed and solved on neural networks. A Hopfield network model [33,34] is used in a new identification structure, for learning time varying and time invariant systems, which consist of mutually inter-connected processing units called neurons. This time domain approach results in system

parameters in terms of activation levels of the network neurons. The second technique, which is in frequency domain, utilizes a set of orthogonal basis functions and fourier analysis network to construct a dynamic system in terms of its fourier coefficients.

Reynold Chu and Manoel Tenorio [35] presented a paper to extend the above idea of system identification. The objective of this paper is to discuss how to apply Hopfield network to the problem of linear system identification. A procedure for programming Hopfield network is presented by measuring inputs and time derivatives of state variables. The states of neurons of this network are guaranteed to converge to system parameters which are to be identified. Hopfield has also shown that his network can solve signal decomposition problems in which the goal is to calculate the optimal fit of an integer coefficient combination of basis functions (possibly a no orthogonal set) to an analogy signal. This paper present show trigonometric functions can be used so that fourier transform of the signal can be generated. As shown, it is simpler to pose this problem in the form of an adaptive linear combiner (ALC), which was originally proposed by Bernard Widrow for his ADALINE Network. The paper describes that by proper selection of learning constants and the sampling time for updating network weights, we can transform the continuous version of Widrow-Hoff rule into Newton's method with only a single step, i.e., weights convergence is achieved in only one update cycle. The convenience provided by this approach is due to the orthogonality of the basis functions. Computer simulations are conducted for two cases: (1) signal decomposition of a flat spectrum with nonzero mean and linear phase delay, (2) identification of the frequency response of a mass-spring damper system subjected to periodic input pulse.

After that, Satyendra Bhama and H Singh [36] presented another technique for system identification In this paper it is described tat how to use the gradient descent (GD) technique with single layer neural networks (SLNN's) to identify the parameters of a linear dynamical system whose states and derivatives of state are given. Gradient descent learning algorithm is also known as instant back-propagation to train a single layer neural network (SLNN) for identifying the parameters of a linear system. An appropriately formulated least mean square error (LMSE) is used as a performance measure for the network proposed. The proposed network is simpler in structure and, therefore, less expensive in implementation. We can use

such a network for dynamic image modeling [37,38]. The circuit is considered to be faster and is recommended for on-line computation because of the parallel nature of its architecture and the possibility of the use of analog circuit components. The computer simulation results for determining the parameters of an unknown linear dynamical system show that circuit is suitable for on-line computation.

It is well known ADALINE is slow in convergence which is not appropriate for online application and identification of time varying system. Two techniques are proposed to speed up convergence of learning, thus increase the capability of tracking time varying system parameters. One idea is to introduce a momentum term to the weight adjustment during convergence period. The other technique is to train the generalized ADALINE network multiple epochs with data from a sliding window of the system's input output data. So, we move to new approach called GADALINE. GADALINE is a simple linear neural network which can be trained to adaptively model a linear or non-linear system.

Wenle Zhang [39,40] presents papers based on a Generalized Adaptive Linear Element (ADALINE) neural network, called GADALINE, for linear time varying systems which can be described by a discrete time model. In such systems, the current system output is dependent on past outputs and on both the current and past inputs. So, the GADALINE needs to have output feedback to remember past input/output data. The proposed neural network employed a tapped delay line (TDL) for both input and output of the system to remember the past input/output data. GADALINE becomes, then, a linear recurrent type of neural network. Here, the adaptive learning is generalized by adding a momentum term to the GADALINE weight adjustment. This momentum term is turned on only during convergence period and the learning curve is, therefore, smoothed by turning off the momentum once the learning error is within a given small number – epsilon. The author further generalizes the online training by using each set of samples for several epochs obtained from a sliding window of the system's input and output data. Simulation results showed that GADALINE provides a much faster convergence speed and better tracking of time varying parameters. Due to the simple structure of the GADALINE, the learning algorithm has an exceptionally low computational complexity and it makes this method suitable for online system identification and real time adaptive control applications. The

momentum term helps to speed up the learning process and reduce the zigzag effect during convergence period of learning. The second training technique uses each set of samples for several epochs obtained from a sliding window of the system's input output data. These simple generalized LMS learning algorithms with very low computational complexity make the GADALINE based identification method suitable for real time application. The increased convergence speed improves the capability of tracking time varying parameters. Simulations results show that with an appropriate momentum constant α value, GADALINE offers a better performance as compared to original LMS learning [41] and the sliding window provides even better performance when the number of epochs and window size are tuned to appropriate values.

Experimental study was, then, performed to search for an optimal combination of the momentum term and the learning rate. The goal was to speed up convergence (or tracking) while keeping smooth tracking during any transient period. Extensive simulation study towards optimizing the momentum term and learning rate parameter is performed. Optimal or near optimal range for momentum term combined with proper learning rate is determined. Simulation results show that once the momentum factor and learning rate are tuned properly, time varying parameters of LTI systems can be identified quite effectively; which, in turn, shows that the fine tuned GADALINE is quite suitable for online system identification and real time adaptive control applications due to its low computational demand.

Next, an average of weight adjustment and dual epoch learning are proposed to improve the performance of GADALINE neural network. Simulation results show that the proposed method provides indeed faster convergence and better tracking of time varying parameters. The author simplifies the sliding window technique by doing a dual epoch on each new data sample and adding average weight adjustment to help to keep learning momentum while reserve smooth learning. Simulation results showed that the improved GADALINE provides faster convergence and better tracking of time varying parameters. Due to the simple structure of the ADALINE, the learning algorithm has an exceptionally low computational complexity and it makes this method suitable for online system identification and real time adaptive control applications

Peter M. Mills and Albert Y. Zomaya [42] presented a paper for system identification of systems with non-linear transfer functions. Industrial processes often include dynamic non-linear transfer functions which may cause problems for linear identification. Dynamic identification for non-linear processes can be achieved using neural networks. Research to date has concentrated on the area of off-line identification where a short history of input-output samples has been available to generate a model. However, for the purpose of adaptive control of non-linear processes, an on-line algorithm is required to provide on-going improvement of identification accuracy and the ability to track process transfer function changes as they occur. This requires both the need to introduce data from the process as well as the ability to discount old data which may no longer be valid. Corresponding techniques in linear adaptive control are already familiar and provide a benchmark for comparison of new neural techniques.

In contrast with previously published work, this study develops on-line capability for neural identification based on the Error Back-Propagation learning algorithm. Its performance is compared with recursive linear identification commonly used for self-tuning control. To show the general applicability of the technique, a complex model is used as the system to be identified. The model is constructed from formidable static and dynamic non-linear components which occur in real industrial process plants.

This study has introduced three new aspects to neural identification of non-linear systems. Firstly, a method of extending the well known Error back-propagation neural network to perform on-line identification of a system is presented. This method enables the investigation of adaptive non-linear process control based on neural identification. Secondly, the neural identification has been successfully tested on a complex non-linear composite system which includes formidable but realistic non-linear process characteristics such as hysteresis. This has helped to demonstrate the general applicability of identification using neural techniques. Thirdly, the new method of neural identification was compared with on-line identification based on the well established linear least-squares technique. The comparison highlights the faster adaption of linear identification against the higher asymptotic accuracy of neural identification. For identification the system input is a Gaussian white noise source generating a new value of u(t) at each simulation sample [43,44].

Neural Networks for Noise Cancellation

Maja Stella, Dinko Begusic, Mladen Russo [45] presented a paper work with several experiments to see how ADALINE performs as adaptive filter in real life scenarios. Adaptive filters have many practical applications and they are used in a variety of fields. Adaptive filters have the ability to adjust their own parameters automatically and their design requires little or no priori knowledge of signal or noise characteristics [25,26]. An adaptive noise canceller adaptively filters a noise reference input to maximally match and subtract out noise or interference from the primary (signal plus noise) input. In this paper, ADALINE neural network is used as adaptive filter to cancel engine noise in a car. Experiments show that SNR improves after passing the noise cancellation system.

BernardWidrow, John M. Mccool [46] present paper which describes the concept of adaptive noise cancelling, an alternative method of estimating signals corrupted by additive noise or interference. The method uses a "primary" input containing the corrupted Signal and a "reference" input containing noise correlated in some unknown way with the primary noise. The reference input is adaptively filtered and subtracted from the primary input to obtain the signal estimate. Adaptive filtering before subtraction allows the treatment of inputs that are deterministic or stochastic, stationary or time variable. Mathematical analysis show that when the reference input is free of signal and certain other conditions are met noise in the primary input can be essentially eliminated without signal distortion. Adaptive noise cancelling is a method of optimal filtering that can be applied whenever a suitable 'reference' input is available. The principal advantages of the method are its adaptive capability, its low output noise, and its low signal distortion. The adaptive capability allows the processing of inputs whose properties are unknown and nonstationary in some cases. It leads to a stable system that automatically turns itself off when no improvement in signal-to-noise ratio can be achieved. Output noise and signal distortion are generally lower than can be achieved with conventional optimal filter configurations. It is further shown that in treating periodic interference the adaptive noise canceller acts as a notch filter with narrow bandwidth, infinite null, and the capability of tracking the exact frequency of the interference; in this case the canceller behaves as a liners, time-

invariant system, with the adaptive filter converging on a dynamic rather than a static solution. The experimental data presented in this paper demonstrate the ability of adaptive noise cancelling greatly to reduce additive periodic or stationary random interference in both periodic and random signals. In each instance cancelling was accomplished with little signal distortion even though the frequencies of the signal and the interference overlapped. Experimental results are presented that illustrate the usefulness of the adaptive noise cancelling technique in a variety of practical applications.

Neural Networks for Adaptive Prediction

The paper presented by Shuitsu Matsumura, Takuji Maezawa, [47] proposes an application based on adaptive prediction approach, in which the least-square-based predictors are designed for both horizontal scanning and vertical scanning for lossless image coding. In typical continuous tone images, the intensities of adjacent pixels are highly correlated. The predictive coding is aimed to remove the redundancies in images. This paper proposes an approach to get higher compression rate by allowing the increase of a little computational complexity. Here, least square (LS) based approach is introduced into image coding, where the optimal coefficients for predictor can be obtained for each image.

David Samek, Peter Dostal [48] shows the benefits of ADALINE in adaptive prediction. The aim of this paper is to present and compare artificial neural networks as interesting way how to model and predict nonlinear systems even with time-variant parameters. This paper has emphasis mainly on computational costs of the selected predictors and usage of adaptive linear network which offers short learning times and remarkable prediction error. The paper shows that ADALINE in the control task showed impressive results, including control quality, fast training efficient prediction times and is adaptive in nature. Artificial neural network (ANN) model is data-driven. Learning ability makes artificial neural networks versatile, user friendly and powerful tool for many practical applications.

Techniques/ methods:

First, Raymond H. Kwong, Edward W. Johnston [49] propose a variable step size LMS algorithm where the learning- rate parameter adjustment is controlled by the square of the prediction error. The motivation is that a large prediction error will cause the step size to increase to provide faster tracking while a small prediction error will result in a decrease in the step size to yield smaller misadjustment. The adjustment equation is simple to implement, and its form is such that a detailed analysis of the algorithm is possible under the standard independence assumptions commonly made in the literature [l] to simplify the analysis of LMS algorithm. This method reduces the tradeoff between misadjustment and tracking ability of the fixed step size LMS algorithm. The variable step size algorithm also reduces sensitivity of the misadjustment to the level of non-stationarity. Comparison of the new algorithm with the fixed step size algorithm and another variable step algorithm shows that it has superior performance to the fixed step size algorithm, and performs well.

Next, V. John Mathews, Zhenhua Xie, [50] presents an adaptive step-size (learning-rate parameter) gradient adaptive filter. The step size (learning- rate parameter) of the adaptive filter is changed according to a gradient descent algorithm, designed to reduce the squared estimation error during each iteration. The algorithm is different from other traditional methods involving time varying learning- rate parameters. Results show that the initial convergence rate of the adaptive filters is very fast. After an initial period when the learning- rate parameter increases very rapidly, the learning- rate parameter decreases slowly and smoothly, giving rise to small misadjustment errors; and in the case of nonstationary environments, this algorithms seeks to adjust the learning- rate parameters in such a way as to obtain close to the best-possible performance. The tracking performance of these algorithms in non-stationary environments is relatively insensitive to the choice of the parameters of the adaptive filter and is very close to the best possible performance. The good properties and the computational simplicity associated with the algorithm makes it successfully in several practical applications.

Next algorithm is proposed by Tyseer Aboulnasr, K. Mayyas [51] which presents a robust variable step-size LMS (learning -rate parameter)-type algorithm provides fast convergence at early stages of adaptation while ensures small final misadjustment. The performance of this algorithm is not affected by existing uncorrelated noise disturbances. An approximate analysis of convergence and steady-state performance for zero-mean stationary Gaussian inputs and for nonstationary optimal weight vector is provided. For nonstationary environments, this algorithm performs equivalent to that of the regular LMS algorithm. Simulation results comparing the proposed algorithm to current variable learning-rate parameter algorithms clearly indicate its superior performance for cases of stationary environments. The learning -rate parameter of the algorithm is adjusted according to the square of a time-averaging estimate of the autocorrelation of e(n) and e(n-1). As a result, the algorithm can effectively sense the proximity to the optimum independent of uncorrelated measurement noise. The performance of the algorithm was compared with that of the standard LMS [40] as well as other variable learning-rate parameter LMS algorithms [49,50] through simulations. Results show that the algorithm has a significant convergence rate improvement over those algorithms in a stationary environment for the same excess MSE in both high and low SNR environments.

CHAPTER 3
NEURAL NETWORKS USING ADALINE & GADALINE

3.1 Neural Network

A Neural Network is a powerful data-modeling tool that is able to capture and represent complex input/output relationships. The motivation for the development of neural network technology stemmed from the desire to develop an artificial system that could perform "intelligent" tasks similar to those performed by the human brain. Neural networks resemble the human brain in the following two ways:

- A neural network acquires knowledge through learning.
- A neural network's knowledge is stored within inter-neuron connection strengths known as synaptic weights. [2]

Synthetic networks that emulate the biological neural networks found in living organisms are called Artificial Neural Networks. Artificial neural networks have undoubtedly been biologically inspired, but the close correspondence between them and real neural systems is still rather weak. A neural network consists of an interconnected group of artificial neurons, and it processes information using a connectionist approach to computation. In most cases an ANN is an adaptive system that changes its structure based on external or internal information that flows through the network during the learning phase. Modern neural networks are non-linear statistical data modeling tools. They are usually used to model complex relationships between inputs and outputs or to find patterns in data.

3.2 Advantages of Neural Networks

Either humans or other computer techniques can use neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, to extract patterns and detect

trends that are too complex to be noticed. A trained neural network can be thought of as an "expert" in the category of information it has been given to analyze. Advantages include:

- Adaptive learning: An ability to learn how to do tasks based on the data given for training or initial experience.
- Self-Organization: An ANN can create its own organization or representation of the information it receives during learning time.
- Real Time Operation: ANN computations may be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability.
- Fault Tolerance via Redundant Information Coding: Partial destruction of a network leads to the corresponding degradation of performance. However, some network capabilities may be retained even with major network damage.

3.3 Neural Networks versus Conventional Computers

Neural networks take a different approach to problem solving than that of conventional computers.

- Conventional computers use an algorithmic approach i.e. the computer follows a set of instructions in order to solve a problem. Unless the specific steps that the computer needs to follow are known the computer cannot solve the problem. That restricts the problem solving capability of conventional computers to problems that we already understand and know how to solve. But computers would be so much more useful if they could do things that we don't exactly know how to do.

  Neural networks on the other hand, process information in a similar way the human brain does. The network is composed of a large number of highly interconnected processing elements (neurons) working in parallel to solve a specific problem. Neural networks learn by example. They cannot be programmed to perform a specific task.

- The disadvantage of neural networks is that because the network finds out how to solve the problem by itself, its operation can be unpredictable.

On the other hand, conventional computers use a cognitive approach to problem solving; the way the problem is to solve must be known and stated in small unambiguous instructions. These instructions are then converted to a high-level language program and then, into machine code that the computer can understand. These machines are totally predictable; if anything goes wrong is due to a software or hardware fault.

Neural networks and conventional algorithmic computers are not in competition but complement each other. There are tasks are more suited to an algorithmic approach like arithmetic operations and tasks that are more suited to neural networks. Even more, a large number of tasks require systems that use a combination of the two approaches (normally a conventional computer is used to supervise the neural network) in order to perform at maximum efficiency.

3.4 Biological vs. Artificial Neural Network

3.4.1 Biological Neural Network

A biological Neural Network is a series of interconnected biological neurons. A biological neuron receives inputs from other sources, combines them in some way, performs a generally nonlinear operation on the result, and then output the final result. Output can be excited or not excited, subject to attenuation in the synapses, which are junction parts of the neuron. Incoming signals from other neurons determine if the neuron shall excite ("fire"). Figure 4 shows neuron structure. The facts about Biological Neural Networks which motivated humans to implement architecture similar to them

- The number of neurons in the human brain: 10^{11}
- The average number of connections of each neuron: 10^{4}
- Highly parallel computation

Much is still unknown about how the brain trains itself to process information, so theories abound. In the human brain, a typical neuron collects signals from others through a host of fine structures called dendrites.

The neuron sends out spikes of electrical activity through a long, thin stand known as an axon, which splits into thousands of branches. At the end of each branch, a structure called a synapse converts the activity from the axon into electrical effects that inhibit or excite the activity in the connected neurons. When a neuron receives excitatory input that is sufficiently large compared with its inhibitory input, it sends a spike of electrical activity down its axon. Learning occurs by changing the effectiveness of the synapses so that the influence of one neuron on another changes [3].

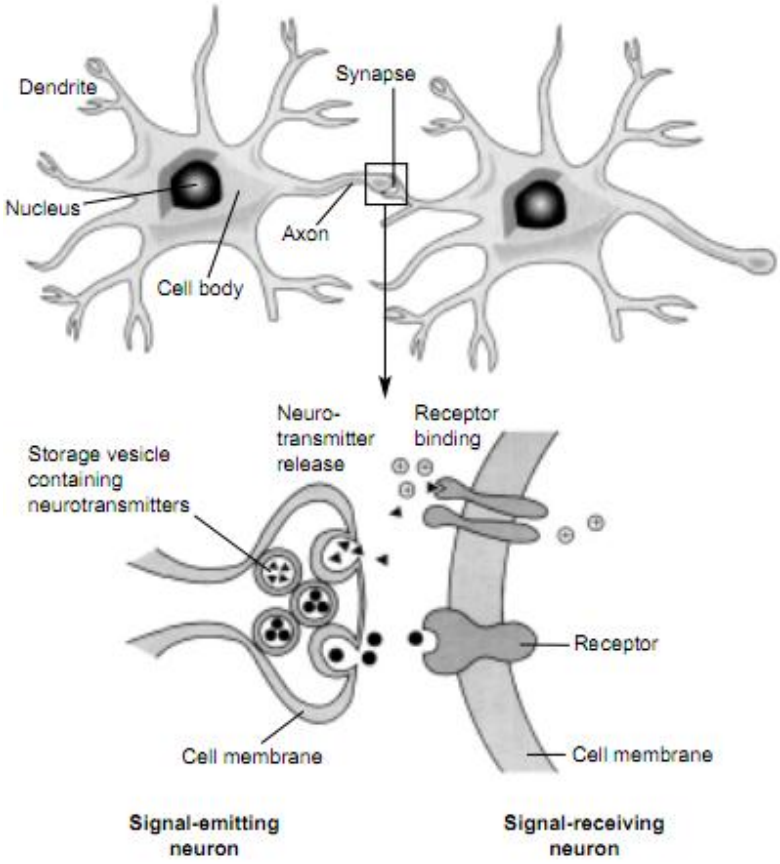

Figure 4 *Neuron structure and Synapse [2]*

Few terms related to the Biological Neural Network:

• **Neuron**: Electrically excitable cell that processes and transmits information by electrical signalling.

• **Dendrites**: Branches of neurons that receive signals from other neurons and pass the signals into the soma.

• **Soma**: Cell body of the neuron.

• **Axons**: The interface through which neurons interact with their neighbouring neurons

• **Synapse**: Electrochemical contact between Neurons.

3.4.2 Human Neurons to Artificial Neurons

We conduct these neural networks by first trying to deduce the essential features of neurons and their interconnections. We then typically program a computer to simulate these features. However because our knowledge of neurons is incomplete and our computing power is limited, our models are necessarily gross idealizations of real networks of neurons. Borrowing from biology, researchers are exploring neural networks—a new, non algorithmic approach to information processing.

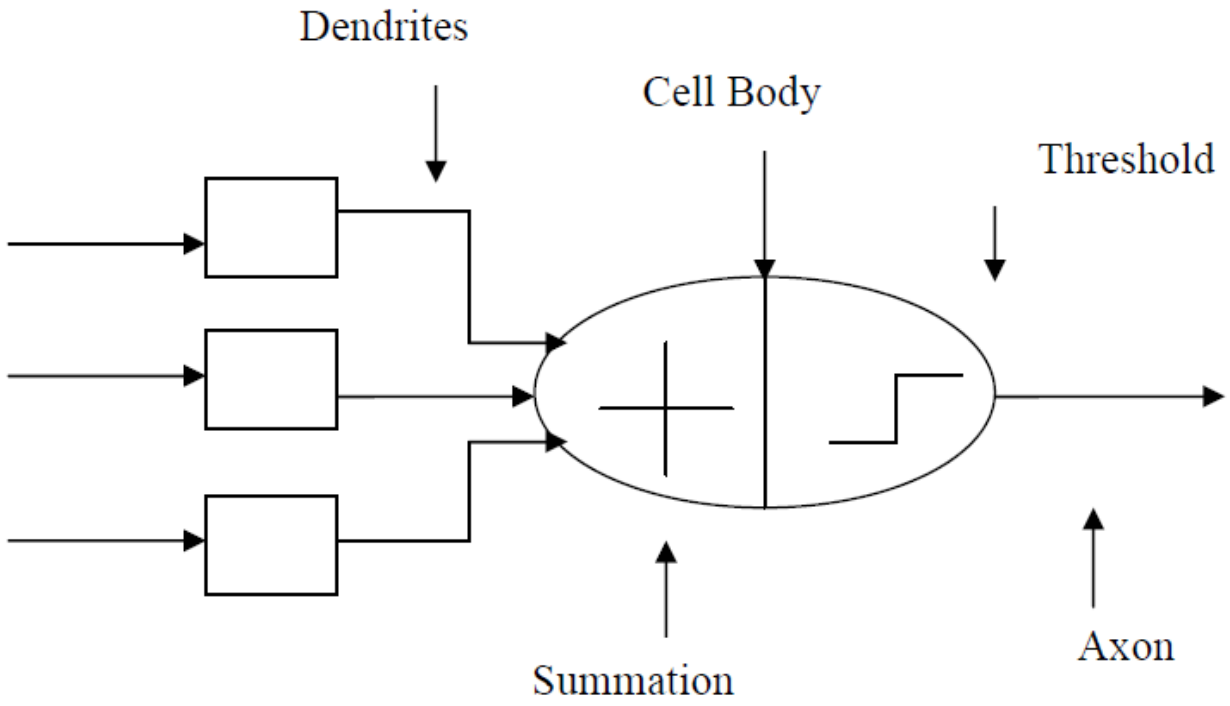

Figure 5 *Neuron Model*

A neural network is a powerful data-modeling tool that is able to capture and represent complex input/output relationships. The motivation for the development of neural network technology stemmed from the desire to develop an artificial system that could perform "intelligent" tasks similar to those performed by the human brain. Neural networks resemble the human brain in the following two ways:

- A neural network acquires knowledge through learning.
- A neural network's knowledge is stored within inter-neuron connection strengths known as synaptic weights.

Artificial Neural Networks are being counted as the wave of the future in computing. They are indeed self-learning mechanisms which don't require the traditional skills of a programmer. But unfortunately, misconceptions have arisen. Writers have hyped that these neuron-inspired processors can do almost anything. These exaggerations have created disappointments for some potential users who have tried, and failed, to solve their problems with neural networks. These application builders have often come to the conclusion that neural nets are complicated and confusing. Unfortunately, that confusion has come from the industry itself. An avalanche of articles has appeared touting a large assortment of different neural networks, all with unique claims and specific examples. Currently, only a few of these neuron-based structures, paradigms actually, are being used commercially. One particular structure, the feed forward, back-propagation network, is by far and away the most popular. Most of the other neural network structures represent models for "thinking" that are still being evolved in the laboratories. Yet, all of these networks are simply tools and as such the only real demand they make is that they require the network architect to learn how to use them [2]. The power and usefulness of artificial neural networks have been demonstrated in several applications including speech synthesis, diagnostic problems, medicine, business and finance, robotic control, signal processing, computer vision and many other problems that fall under the category of pattern recognition. For some application areas, neural models show promise in achieving human-like performance over more traditional artificial intelligence techniques.

3.5 History of Neural Networks

The study of the human brain is thousands of years old. With the advent of modern electronics, it was only natural to try to harness this thinking process [6].
The history of neural networks that was described above can be divided into several periods:

- **First Attempts:** There were some initial simulations using formal logic. McCulloch and Pitts (1943) developed models of neural networks based on their understanding of neurology.

These models made several assumptions about how neurons worked. Their networks were based on simple neurons which were considered to be binary devices with fixed thresholds. The results of their model were simple logic functions such as "a or b" and "a and b". Another attempt was by using computer simulations. Two groups (Farley and Clark, 1954; Rochester, Holland, Haibit and Duda, 1956).The first group (IBM researchers) maintained closed contact with neuroscientists at McGill University. So whenever their models did not work, they consulted the neuro scientists. This interaction established a multidisciplinary trend which continues to the present day.

- **Promising & Emerging Technology**: Not only was neuroscience influential in the development of neural networks, but psychologists and engineers also contributed to the progress of neural network simulations. Rosenblatt (1958) stirred considerable interest and activity in the field when he designed and developed the perceptron. The perceptron had three layers with the middle layer known as the association layer. This system could learn to connect or associate a given input to a random output unit. Another system was the ADALINE (Adaptive Linear Element) which was developedin 1960 by Widrow and Hoff (of Stanford University). The ADALINE was analogue electronic device made from simple components. The method used for learning was different to that of the perceptron; it employed the least mean square (LMS) learning rule.

- **Period of Frustration & Disrepute:** In 1969 Minsky and Papert wrote a book in which they generalized the limitations of single layer perceptrons to multilayered systems. In the book they said: "...our intuitive judgment that the extension (to multilayer systems) is sterile". The significant result of their book was to eliminate funding for research with neural network simulations. The conclusions supported the disenchantment of researchers in the field. As a result, considerable prejudice against this field was activated.

- **Innovation**: Although public interest and available funding were minimal, several researchers continued working to develop neuro morphically based computational methods for problems such as pattern recognition. During this period several paradigms were generated which modern work continues to enhance. Grossberg's (Steve Grossberg and Gail

Carpenter in 1988) influence founded a school of thought which explores resonating algorithms. They developed the ART (Adaptive Resonance Theory) networks based on biologically plausible models. Anderson and Kohonen developed associative techniques independent of each other. Klopf (A. Henry Klopf) in 1972 developed a basis for learning in artificial neurons based on a biological principle for neuronal learning called heterostasis. Werbos (Paul Werbos 1974) developed and used the back-propagation learning method, however several years passed before this approach was popularized. Backpropagation nets are probably the most well known and widely applied of the neural networks today. In essence, the back-propagation net. is a Perceptron with multiple layers, a different threshold function in the artificial neuron, and a more robust and capable learning rule. Amari (A. Shun-Ichi 1967) was involved with theoretical developments: he published a paper which established a mathematical theory for a learning basis (error-correction method) dealing with adaptive pattern classification. While Fukushima (F. Kunihiko) developed a step wise trained multilayered neural network for interpretation of handwritten characters. The original network was published in 1975 and was called the cognitron.

- **Re-Emergence:** Progress during the late 1970s and early 1980s was important to the re-emergence on interest in the neural network field. Several factors influenced this movement. For example, comprehensive books and conferences provided a forum for people in diverse fields with specialized technical languages, and the response to conferences and publications was quite positive. The news media picked up on the increased activity and tutorials helped disseminate the technology. Academic programs appeared and courses were introduced at most major Universities (in US and Europe). Attention is now focused on funding levels throughout Europe, Japan and the US and as this funding becomes available, several new commercial with applications in industry and financial institutions are emerging.

- **Today:** Significant progress has been made in the field of neural networks-enough to attract a great deal of attention and fund further research. Advancement beyond current commercial applications appears to be possible, and research is advancing the field on many fronts. Neurally based chips are emerging and applications to complex problems developing. Clearly, today is a period of transition for neural network technology.

3.6 An Engineering Approach

3.6.1 A Simple Neuron

An artificial neuron is a device with many inputs and one output. The neuron has two modes of operation, the training mode and the using mode.

- In the training mode, the neuron can be trained to fire (or not), for particular input patterns.
- In the using mode, when a taught input pattern is detected at the input, its associated output becomes the current output. If the input pattern does not belong in the taught list of input patterns, the firing rule is used to determine whether to fire or not.

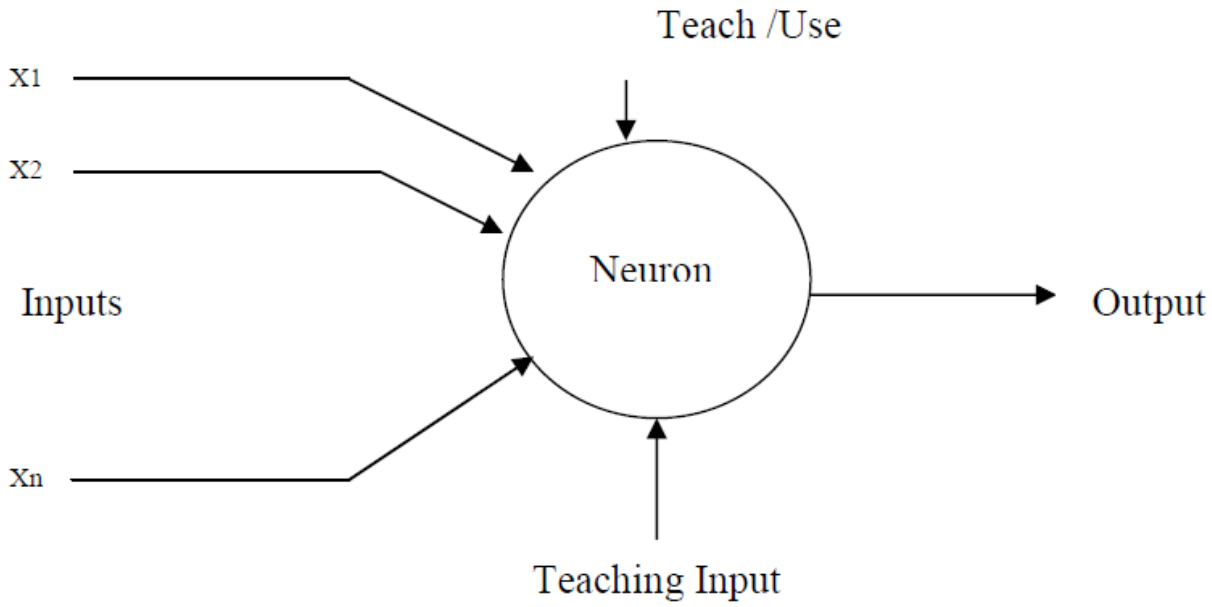

Figure 6 *A Simple Neuron*

3.6.2 Firing Rules

The firing rule is an important concept in neural networks and accounts for their high flexibility. A firing rule determines how one calculates whether a neuron should fire for any input pattern. It relates to all the input patterns, not only the ones on which the node was trained. A simple firing rule can be implemented by using Hamming distance technique. The rule goes as:

Take a collection of training patterns for a node, some of which cause it to fire (the 1 taught set of patterns) and others, which prevent it from doing so (the 0-taught set). Then the patterns not in the collection cause the node to fire if, on comparison, they have more input elements in common with the 'nearest' pattern in the 1-taught set than with the 'nearest' pattern in the 0-taught set. If there is a tie, then the pattern remains in the undefined state.

For example, a 3-input neuron is taught to output 1 when the input (X1, X2 and X3) is111 or 101 and to output 0 when the input is 000 or 001.
Then, before applying the firing rule, the truth table is:

X1:	0	0	0	0	1	1	1	1
X2:	0	0	1	1	0	0	1	1
X3:	0	1	0	1	0	1	0	1
OUT:	0	0	0/1	0/1	0/1	1	0/1	1

Table 1 *Truth Table before applying the firing rule*

As an example of the way the firing rule is applied, take the pattern 010. It differs from 000 in 1 element, from 001 in 2 elements, from 101 in 3 elements and from 111 in 2 elements. Therefore, the 'nearest' pattern is 000, which belongs, in the 0-taught set. Thus the firing rule requires that the neuron should not fire when the input is 001. On the other hand, 011 is equally distant from two taught patterns that have different outputs and thus the output stays undefined (0/1).
By applying the firing in every column the following truth table is obtained:

X1:		0	0	0	0	1	1	1	1
X2:		0	0	1	1	0	0	1	1
X3:		0	1	0	1	0	1	0	1
OUT:		0	0	0	0/1	0/1	1	1	1

Table 2 *Truth Table after applying the firing rule*

The difference between the two truth tables is called the generalization of the neuron. Therefore the firing rule gives the neuron a sense of similarity and enables it to respond 'sensibly' to patterns not seen during training.

3.6.3 A More Complicated Neuron

The previous neuron doesn't do anything that conventional computers don't do already. A more sophisticated neuron is the McCulloch and Pitts model (MCP). The difference from the previous model is that the inputs are ‘weighted’; the effect that each input has at decision-making is dependent on the weight of the particular input. The weight of an input is a number which when multiplied with the input gives the weighted input. These weighted inputs are then added together and if they exceed a pre-set threshold value, the neuron fires. In any other case the neuron does not fire.

In mathematical terms, the neuron fires if and only if;

$$X_1W_1 + X_2W_2 + X_3W_3 + ... > T$$

The addition of input weights and of the threshold makes this neuron a very flexible and powerful one. The MCP neuron has the ability to adapt to a particular situation by changing its weights and/or threshold.

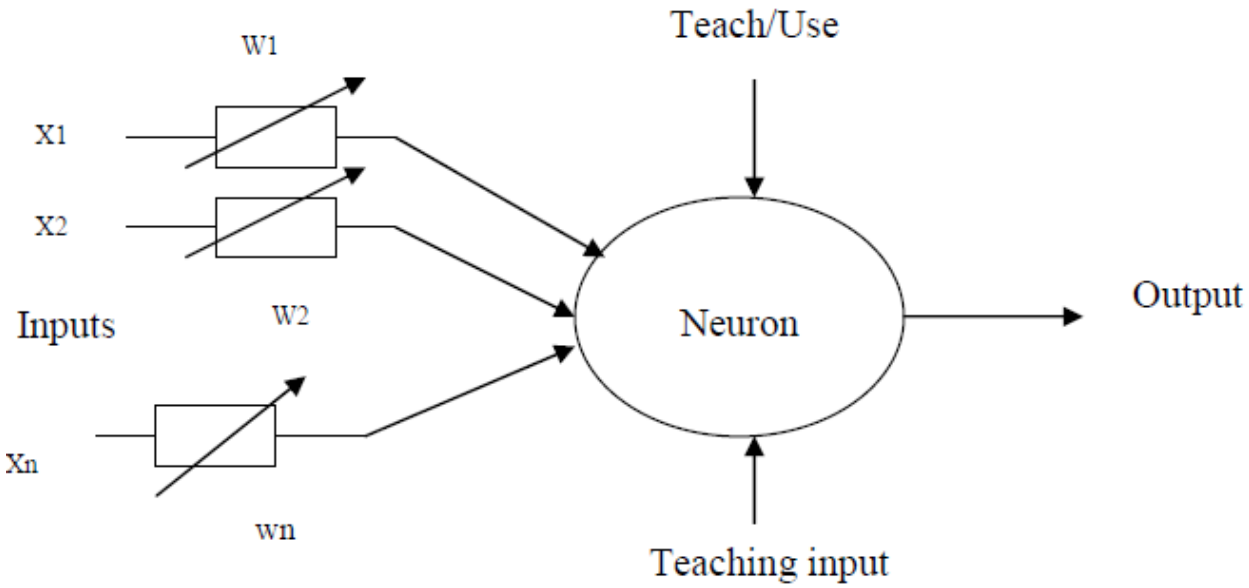

Figure 7*MCP Neuron*

3.7 Architecture of Neural Networks

3.7.1 Network Layers

The commonest type of artificial neural network consists of three groups, or layers, of units: a layer of "input" units is connected to a layer of "hidden" units, which is connected to a layer of "output" units [6].

- The activity of the input units represents the raw information that is fed into the network.
- The activity of each hidden unit is determined by the activities of the input units and the weights on the connections between the input and the hidden units.
- The behavior of the output units depends on the activity of the hidden units and the weights between the hidden and output units.

This simple type of network is interesting because the hidden units are free to construct their own representations of the input. The weights between the input and hidden units determine when each hidden unit is active, and so by modifying these weights, a hidden unit can choose what it

represents. We also distinguish single-layer and multi-layer architectures. The single-layer organization, in which all units are connected to one another, constitutes the most general case and is of more potential computational power than hierarchically structured multilayer organizations. In multi-layer networks, layer, instead of following a global numbering, often numbers units.

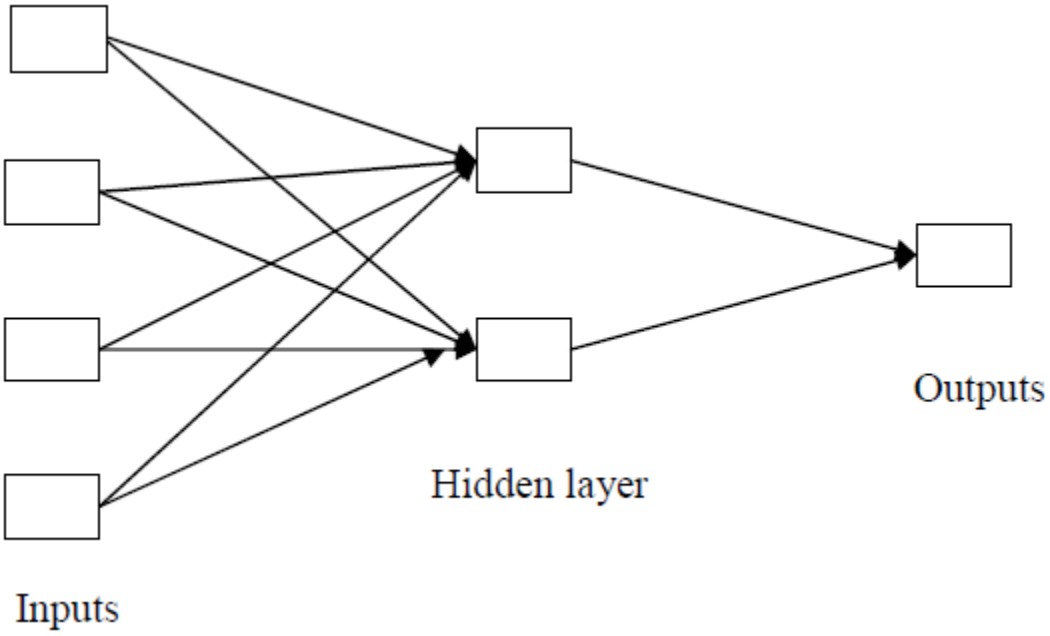

Figure 8 *Network Layers*

3.7.2 Feed-Forward Networks

Feed-forward Ann's allow signals to travel one way only; from input to output. There is no feedback (loops) i.e. the output of any layer does not affect that same layer. Feed-forward Ann's tend to be straightforward networks that associate inputs with outputs. They are extensively used in pattern recognition. This type of organization is also referred to as bottom-up or top-down.

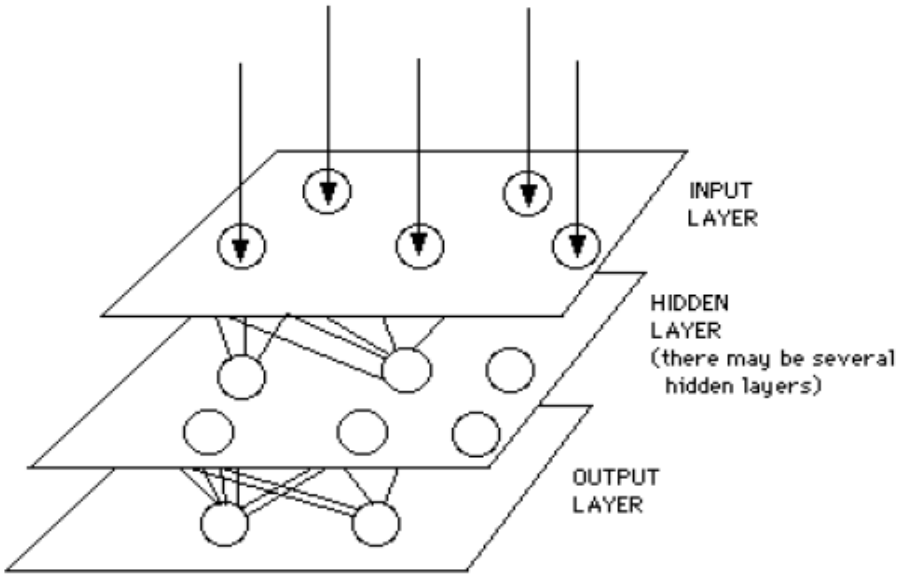

Figure 9*Feed Forward Neural Network*

3.7.3 Feedback Networks

Feedback networks can have signals traveling in both directions by introducing loops in the network. Feedback networks are very powerful and can get extremely complicated. Feedback networks are dynamic; their 'state' is changing continuously until they reach an equilibrium point. They remain at the equilibrium point until the input changes and a new equilibrium needs to be found. Feedback architectures are also referred to as interactive or recurrent, although the latter term is often used to denote feedback connections in single-layer organizations.

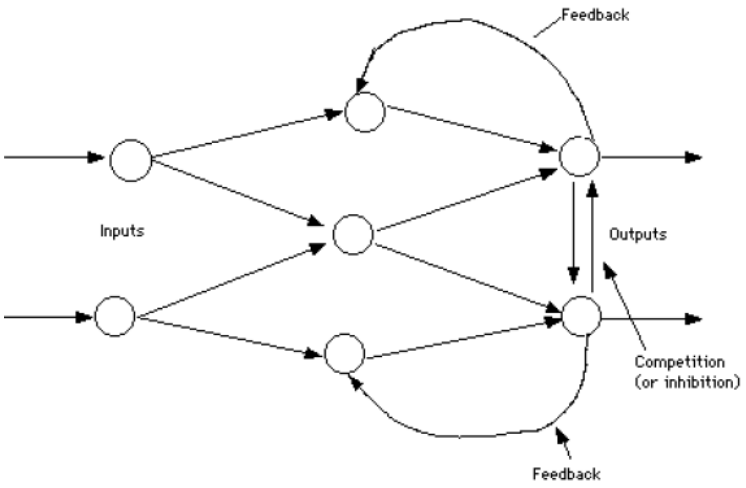

Figure 10 *Feedback Neural Network*

3.8 Perceptrons

The perceptron calculates a weighted sum of inputs and compares it to a threshold. If the sum is higher than the threshold, the output is set to 1, otherwise to -1, depending upon activation function.

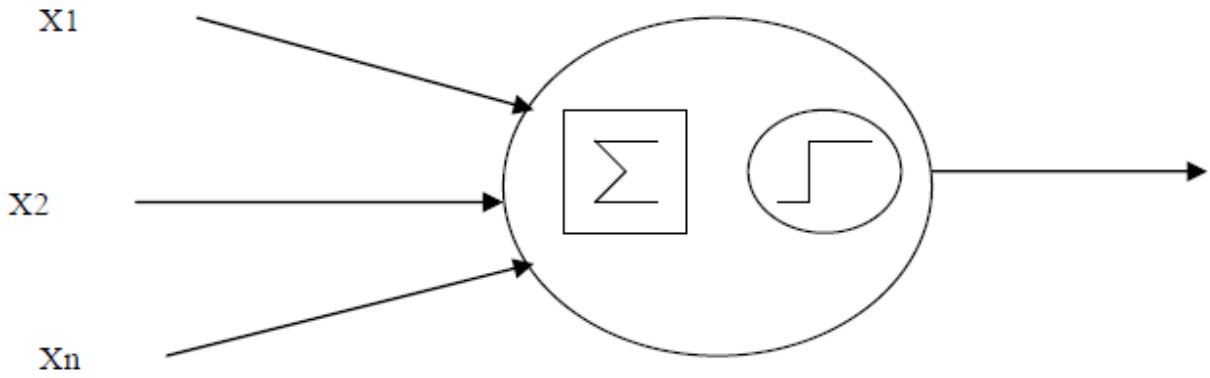

Figure 11 *The Mcculloh-Pitts Model*

In 1969 Minsky and Papert wrote a book in which they described the limitations of single layer Perceptrons. The impact that the book had was tremendous and caused a lot of neural network researchers to loose their interest. The book was very well written and showed mathematically that single layer perceptrons could not do some basic pattern recognition operations like determining the parity of a shape or determining whether a shape is connected or not. What they did not realize, until the 80's, is that given the appropriate training, multilevel perceptrons can do these operations.

3.8.1 The Learning Process

Learning is a process by which the free parameters of a neural network are adapted through a process of stimulation by the environment in which the network is embedded. The type of learning is determined by the manner in which the parameter changes take place [1]. The memorization of patterns and the subsequent response of the network can be categorized into two general paradigms:

ASSOCIATIVE MAPPING in which the network learns to produce a particular pattern on the set of output units whenever another particular pattern is applied on the set of input units. The associative mapping can generally be broken down into two mechanisms:

- Auto-Association: an input pattern is associated with itself and the states of input and output units coincide. This is used to provide pattern completion, i.e. to produce a pattern whenever a portion of it or a distorted pattern is presented. In the second case, the network actually stores pairs of patterns building an association between two sets of patterns [2].

- Hetero-Association: is related to two recall mechanisms
  - Nearest-neighbor recall, where the output pattern produced corresponds to the input pattern stored, which is closest to the pattern presented, and
  - Interpolative recall, where the output pattern is a similarity dependent interpolation of the patterns stored corresponding to the pattern presented.

REGULARITY DETECTION in which units learn to respond to particular properties of the input patterns. Whereas in associative mapping the network stores the relationships among patterns, in regularity detection the response of each unit has a particular 'meaning'. This type of learning mechanism is essential for feature discovery and knowledge representation.

Every neural network possesses knowledge, which is contained in the values of the connections weights. Modifying the knowledge stored in the network as a function of experience implies a learning rule for changing the values of the weights. Information is stored in the weight matrix of

a neural network. Learning is the determination of the weights. Following the way learning is performed; we can distinguish two major categories of neural networks:

- FIXED NETWORKS in which the weights cannot be changed, i.e. dw/dt=0. In such networks, the weights are fixed a priori, according to the problem to solve.

- ADAPTIVE NETWORKS which are able to change their weights, i.e. dw/dt not= 0.

All learning methods used for adaptive neural networks can be classified into two major categories:

1) SUPERVISED LEARNING which incorporates an external teacher, so that each output unit is told what its desired response to input signals ought to be. During the learning process global information may be required. Paradigms of supervised learning include error-correction learning (back propagation algorithm), reinforcement learning and stochastic learning. An important issue concerning supervised learning is the problem of error convergence, i.e. the minimization of error between the desired and computed unit values. The aim is to determine a set of weights which minimizes the error. One well-known method, which is common to many learning paradigms, is the least mean square (LMS) convergence.

a) THE BACK-PROPAGATION LEARNING: A back propagation neural network uses a feed-forward topology, supervised learning, and back propagation learning algorithm. This algorithm was responsible in large part for the re-emergence of neural networks in the mid 1980s [52,53]. Back propagation is a general purpose learning algorithm. It is powerful but also expensive in terms of computational requirements for training. A back propagation network with a single hidden layer of processing elements can model any continuous function to any degree of accuracy (given enough processing elements in the hidden layer).

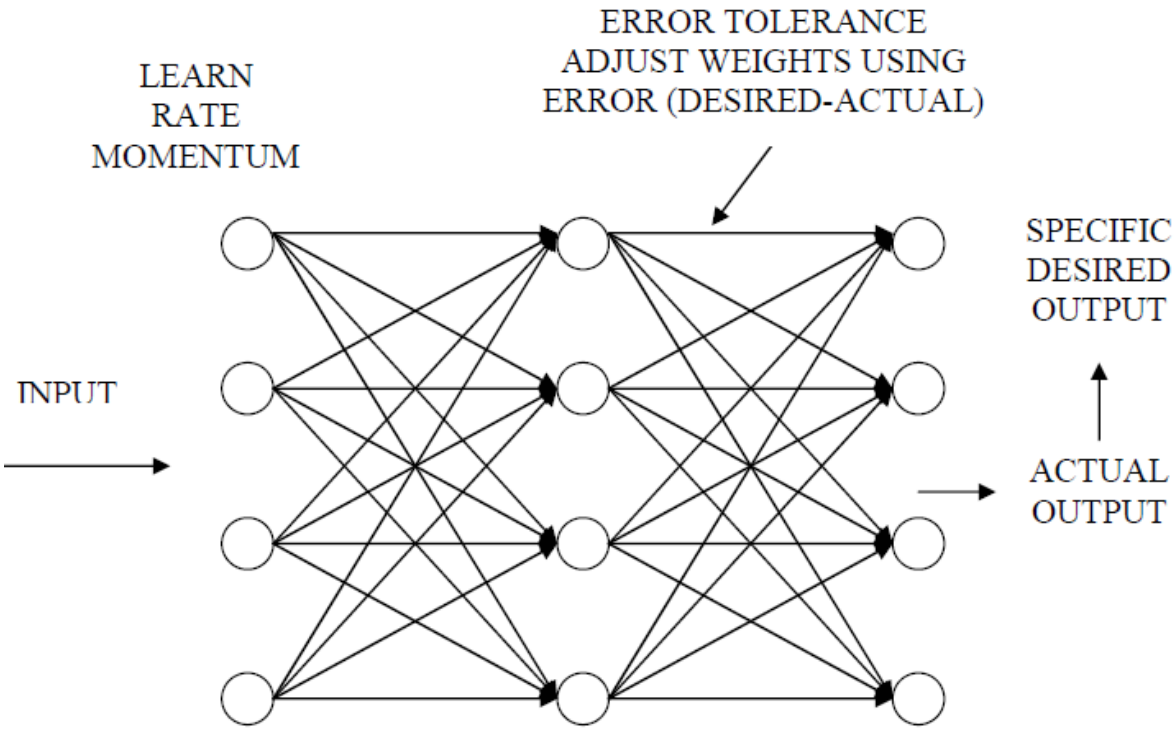

Figure 12 *Back Propagation Network*

There are literally hundreds of variations of back propagation in the neural network literature, and all claim to be superior to "basic" back propagation in one way or the other. Indeed, since back propagation is based on a relatively simple form of optimization known as gradient descent, mathematically astute observers soon proposed modifications using more powerful techniques such as conjugate gradient and Newton's methods.

However, "basic" back propagation is still the most widely used variant. Its two primary virtues are that it is simple and easy to understand, and it works for a wide range of problems. The basic back propagation algorithm consists of three steps.

- The input pattern is presented to the input layer of the network. These inputs are propagated through the network until they reach the output units. This forward pass produces the actual or predicted output pattern.
- Because back propagation is a supervised learning algorithm, the desired outputs are given as part of the training vector. The actual network outputs are subtracted from the desired outputs and an error signal is produced.

- This error signal is then the basis for the back propagation step, whereby the errors are passed back through the neural network by computing the contribution of each hidden processing unit and deriving the corresponding adjustment needed to produce the correct output. The connection weights are then adjusted and the neural network has just “learned” from an experience.

  Two major learning parameters are used to control the training process of a back propagation network. The ***learning-rate parameter*** is used to specify whether the neural network is going to make major adjustments after each learning trial or if it is only going to make minor adjustments. ***Momentum*** is used to control possible oscillations in the weights, which could be caused by alternately signed error signals. While most commercial back propagation tools provide anywhere from 1 to 10 or more parameters for you to set, these two will usually produce the most impact on the neural network training time and performance.

b) REINFORCEMENT LEARNING: Reinforcement learning is a form of supervised learning because the network gets some feedback from its environment. The feedback signal (yes/no reinforcement signal) is only evaluative, not instructive, i.e. if the reinforcement signal says that a particular output is wrong and it gives no hint as to what the right answer should be. It is therefore important in a reinforcement learning network to implement some source of randomness in the network, so that the space of possible outputs can be explored until a correct value is found. Reinforcement learning is sometimes called "learning with a critic" as opposed to "learning with a teacher" which refers to more traditional learning schemes were an error signal is generated which also contain information in which direction the synaptic weights of the networks should be changed in order to improve the performance. In reinforcement learning problems it is common to think explicitly of a network functioning in an environment [6]. The environment supplies the inputs to the network, receives its output and then provides the reinforcement signal.

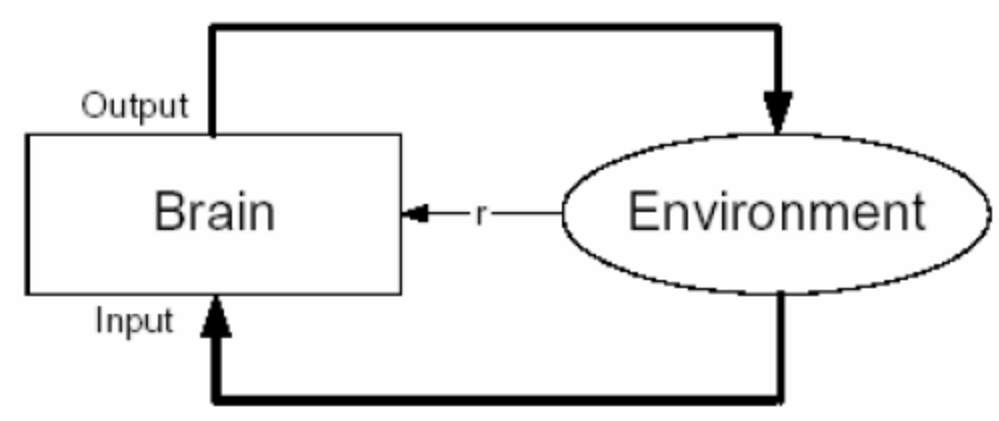

Figure 13 *The Principle layout for a reinforcement learning agent*

2) UNSUPERVISED LEARNING uses no external teacher and is based upon only local information. It is also referred to as self-organization, in the sense that it self-organizes data presented to the network and detects their emergent collective properties. Paradigms of unsupervised learning are hebbian learning and competitive learning.

a) Hebbian learning: The synapse resistance to the incoming signal can be changed during a "learning" process, following quoted by Donald Olding Hebbin his book “The Organization Behavior” [1949], later known as Hebb’s Rule:

Let us assume that the persistence or repetition of a reverberatory activity (or "trace") tends to induce lasting cellular changes that add to its stability.… When an axon of cell A is near enough to excite cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased [3].

b) Competitive Learning: In competitive learning, the output neurons of a neural network compete among themselves to become active (fired) whereas in a neural network based on hebbian learning several output neurons may be active simultaneously, in competitive learning only a single output neuron is active at only one time [6]. There are three basic elements to a competitive learning rule:

- A set of neurons that are all same except for some randomly distributed synaptic weights, and which therefore respond differently to a given set of input patterns.

- A limit imposed on the strength of each neuron.
- A mechanism that permits the neurons to compete for the right to respond to a given subset of inputs, such that only one output neuron, or only one neuron per group is active at a time. The neuron that wins the competition is called a winner takes all neuron.

3.8.2 Transfer Function

The behavior of an ANN (Artificial Neural Network) depends on both the weights and the input-output function (transfer function) that is specified for the units. This function typically falls into one of three categories:

- Linear (or ramp): the output activity is proportional to the total weighted output.
- Threshold: the output is set at one of two levels, depending on whether the total input is greater than or less than some threshold value.
- Sigmoid: the output varies continuously but not linearly as the input changes. Sigmoid units bear a greater resemblance to real neurons than do linear or threshold units, but all three must be considered rough approximations.

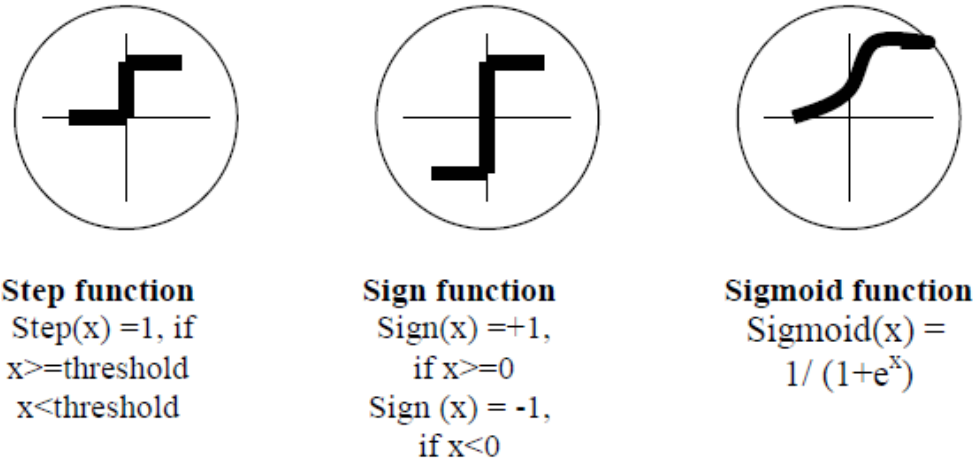

Figure 14 *Types of Transfer Functions*

To make a neural network that performs some specific task, we must choose how the units are connected to one another, and we must set the weights on the connections appropriately. The connections determine whether it is possible for one unit to influence another. The weights

specify the strength of the influence. A three-layer network can be taught to perform a particular task by using the following procedure:

- The network is presented with training examples, which consist of a pattern of activities for the input units together with the desired pattern of activities for the output units.
- It is determined how closely the actual output of the network matches the desired output.
- The weight of each connection can be changed so that the network produces a better approximation of the desired output.

3.9The ADALINE Neural Configuration

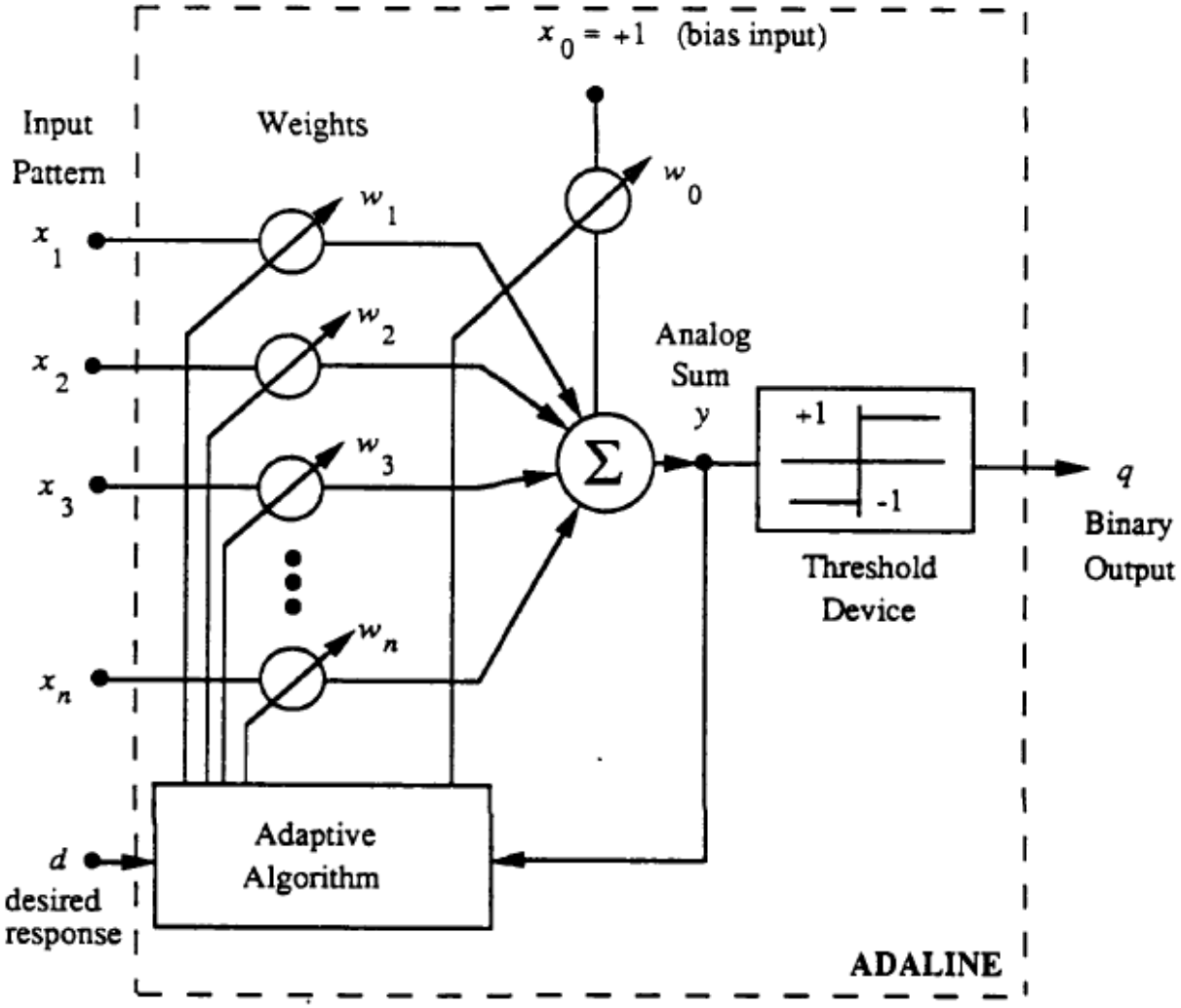

Figure 15 *The Adaptive Linear Neuron, or ADALINE [54]*

The ADALINE is an adaptive threshold logic unit and is depicted in Figure 15**.** The ADALINE performs a weighted sum of its inputs and makes a binary decision based on this sum. The ADALINE's input pattern has *n* variable components. The input pattern together with the constant **+1** bias input form the (n + 1)-dimensional input vector, $\underline{\mathbf{X}} = [x_0=1, x_1 \ldots\ldots x_n]^T$**.** The weighting vector is $\underline{\mathbf{W}} = [w_o, w_1, \ldots, \ldots w_n]^T$ **.**The weighted sum, referred to as the ADALINE's analog sum or analog output, is the dot product of these two vectors, $y = \underline{X}^T\underline{W}$. The ADALINE's binary output results from passing the analog sum through a threshold device, q = sgn(y), where sgn(.) is +1 for positive argument, and-1otherwise.

The individual components of the input vector could have analog values but are often restricted to binary values. Except as noted, the input component values will be either +1 or - 1 in this thesis. There are two reasons for making this restriction. First, the inputs will be binary when an ADALINE receives its inputs from the outputs of other ADALINEs. Second, with this restriction on the inputs, the magnitude of the input vector, $|\underline{X}|$, will be a constant $(n+1)^{1/2}$. The weights are allowed to be continuous, real-valued numbers and are adjusted by an adaptive algorithm.

The ADALINE makes binary decisions based upon its inputs and weights. The nature of the ADALINE's decision-making capability can be determined by examining its governing equation at the decision threshold, that is, when the analog sum y is zero. The equation that describes the decision separating surface for an ADALINE is a dot product relation,

$$Y = |\underline{X}^T\underline{W}| = |\underline{W}^T\underline{X}| = 0 \qquad (3.1)$$

The $\underline{X}$ and $\underline{W}$ vectors can be considered to be located in a continuous, real-valued, (n+1) dimensional vector space. Within this space, the equation above specifies a normality condition. The symmetry of the dot product allows two perspectives in understanding the ADALINE.

In the first perspective, consider the weight vector to be fixed in the input vector space. The decision surface is a hyper plane through the origin and perpendicular to the weight vector. It divides the input vector space in half. Those input vectors lying on the same side of the hyper plane as the weight vector form positive dot products with the weight vector and result in +1

decision by the ADALINE. Those input vectors on the opposite side of the hyper plane result in -1 decision. The perspective, then, is that the weight vector defines a division of the input space into decision regions.

The other perspective involves choosing an input vector in the weight space. The input vector has a desired response associated with it. The hyper plane through the origin and perpendicular to the input vector divides the weight space in half. Weight vectors on only one side of the hyper plane provide the correct desired response for this particular input vector. If now a second input vector is considered, its desired response will also define a half-space where the weight vector must lie to provide a correct response. To accommodate both input vectors, the weight vector must lie in the intersection of the two half-spaces. To satisfy the requirements for an entire training set, the weight vector must lie in the intersection of the half-spaces defined by each input vector and its associated desired response. This intersection is a convex cone emanating from the origin. Any weight vector lying in this cone will be a solution vector for the given training set. The intersection of the half-spaces may be empty. This means there is no weight vector that will provide the desired separation of the input patterns. This second perspective then is that of the input vectors defining a solution region for the weight vector.

The second perspective makes it clear that a single ADALINE cannot perform all the possible separations of the inputs. For input patterns having *n* components, there are 2^n different patterns. A general logic implementation would be capable of classifying each pattern as +1 or -1, in accordance with the desired response. Thus, there are 2^{2^n} possible logic functions connecting *n* inputs to a single output. The single ADALINE can realize only a small subset of these functions, known as the linearly separable functions.

When the dimension of the input vectors is small, graphical methods can illustrate the decision-making capability of the ADALINE. For the case of an ADALINE with two variable inputs, the equation at the decision threshold is:

$$y = w_0 + x_1 w_1 + x_2 w_2 = 0 \qquad (3.2)$$

Rewritten, it becomes:

$$x_2 = - ((w_1/w_2)*x_1+w_0/w_2) \quad (3.3)$$

The plot of the equation is a straight line in the $\mathbf{x_1}$-$\mathbf{x_2}$ input pattern space. The input pattern space is a subspace of the input vector space. It lies in a hyper plane of the input vector space, specifically, the hyper plane defined by x_0= + 1. In the case developed here, the input vector and the weight vector are 3-dimensional but the input pattern is 2-dimensional. The decision hyper plane is a 2-dimensional plane and so is the hyper plane containing the pattern space. Both of these planes exist in 3-space, the space in which the input vectors and the weight vector exist.

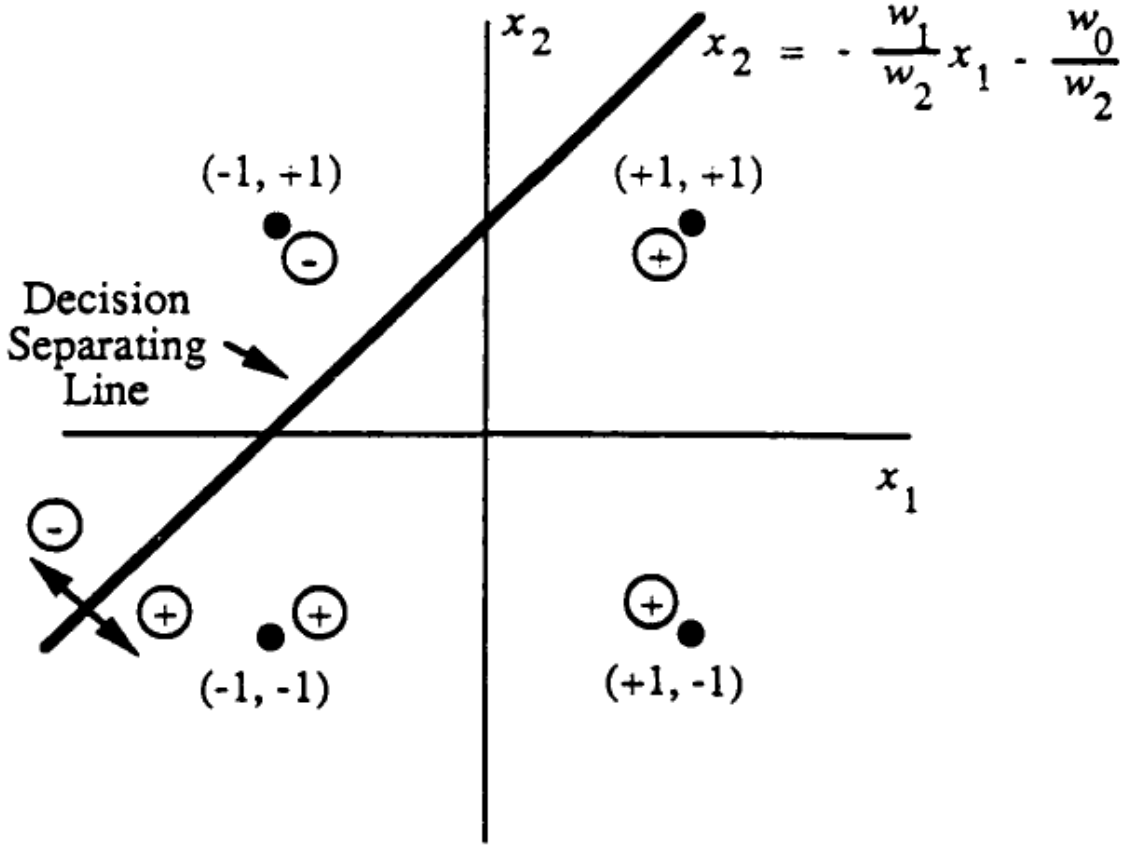

Figure 16 *Graph of ADALINE decision separating line in input 2-space[54]*

The decision hyper plane intersects the pattern space in a line. In the pattern space, the line has a slope of $-w_1/w_2$ and x_2-intercept of $-w_O/w_2$. A sample plot of this line in the pattern space is shown in Figure 16. The line is a decision line. It divides the input patterns into two sets. Inputs

to the right of the separating line cause the ADALINE's analog sum to be positive, resulting in +1decisionbythe ADALINE. Inputs to the left of the line result in -1decisions. The ADALINE thus performs a mapping from its two-dimensional input pattern space to its one-dimensional binary output space. The input/output mapping affected in the example of Figure 16 is represented by**:**

(+1 +1)→+1

(+1 -1)→+1

(-1 -1) →+1

(-1 +1) →-1

By adjusting the weights, the slope and intercept of the line in Figure 16 can be changed. Reversing the signs of all the weights will give exactly the same separating line but will change which side of the line results in a positive decision. Therefore, changing the weights can cause a change in the input/output mapping of the ADALINE. Methods for adapting the weights of a single ADALINE to achieve a desired input/output mapping were developed in the early 1960's.

The mechanism for changing the weights is represented by the "adaptive algorithm" block in Figure 15. The algorithm's inputs are the desired response, d, for the pattern being presented and the actual analog response of the ADALINE. Mays presents three adaptation procedures for the single ADALINE [55]. All three of these procedures produce a set of weights that provide the desired input/output mapping, if such weights exist, in a finite number of adaptations. An example of an input/output mapping that cannot be achieved by a single ADALINE will be shown later.

Mays' formulation of the adaptation procedures includes a concept called the dead zone. In any hardware implementation of an ADALINE, a real thresholding element will be used. If the analog sum y is very close to zero, it is possible that the thresholding element could change output states in an erratic fashion due to noise. Manufacturing tolerances might also cause the actual threshold to be different from zero. For these reasons it is desired that the magnitude of the analog sum be greater than a positive dead zone value delta. The magnitude of the analog sum is called the "confidence level." The confidence level is a measure of how sure the ADALINE is

about its decision. During training then, not only must the binary decision be correct, but the confidence level must be greater than the dead zone value.

All of the ADALINE adaptation procedures can be summarized as follows. There exists a set of patterns and associated desired responses that the ADALINE is to learn. This set is called the training set. Present a pattern and its associated desired response from the training set to the ADALINE. If the binary response is correct and the confidence level is greater than the dead zone value, go on to the next pattern. If the response is incorrect or the confidence level is less than the dead zone value, make a change in the weights. The weights are changed by adding or subtracting some portion of the input. The weight changing method that Mays called the modified relaxation procedure is detailed as follows:

$$\underline{W}(k+1) = \underline{W}(k)$$

for $d(k)\underline{X}(k)^T\underline{W}(k) > \text{delta}$

$$\underline{W}(k+1) = \underline{W}(k) + \eta\, d(k)/(n+1)[L - d(k)\underline{X}(k)\underline{W}(k)] \qquad (3.4)$$

for $d(k)\underline{X}(k)^T\underline{W}(k) < \text{delta}$

Here, k is an iteration cycle number so that $\underline{W}(k+1)$ is the weight vector after the k^{th} adaptation. As before, d(k)is the desired binary response, +1or -1**,** associated with the input vector for the k^{th} iteration, $\underline{X}(k)$.The adaptation constant, η **,**must have value $0 < \eta <= 2$ to insure convergence. The dead zone can be selected $0 < \eta < 1$.The quantity L is called the adaptation level and is selected such that $\eta < L$. Generally, L is selected to be + 1.The adaptation level can be thought of as a target confidence level when η = 1**.** To see this, compute the resulting analog sum after adaptation by pre multiplying both side of Equation 3.4 by $\underline{X}^T(k)$,remembering that

$$y(k) = \underline{X}^T(k)\underline{W}(k), \text{ and } \underline{X}^T(k)\underline{X}(k) = n + 1.$$

$$y(k+1) = y(k) + \eta\, d(k)[L - d(k)y(k)] = y(k)[1 - \eta] + \eta\, d(k)L = d(k)L \text{ for } \eta = 1.$$

After adaptation, the ADALINE will provide the correct response with confidence equal to the adaptation level when η =1 is used in the modified relaxation scheme.

3.10 The Generalized ADALINE

ADALINE (Widrow and Lehr 1990) was first developed to recognize binary patterns so that if it was reading streaming bits from a phone line, it could predict the next bit. MADALINE was the first neural network applied to a real world problem, using an adaptive filter that eliminates echoes on phone lines. The names come from their use of Multiple Adaptive Linear Elements. The structure of an ADALINE is shown in Fig. 15, where threshold logic unit can also be attached to the output. For applying the ADALINE to system identification, we need to modify both the structure and the learning algorithm.

3.10.1 Structure of GADALINE

In the GADALINE structure, the bias input is removed, and each input is expanded through a TDL to become a number of delayed inputs and the output feedback is introduced to the input, also passing through a TDL to become a number of inputs. If we still keep the notation for inputs as $u1 \sim um$, then the output y can be found by,

$$y = \sum w_i u_i = \boldsymbol{u}^T \boldsymbol{w}$$

Where $\boldsymbol{u}$ is the input vector and $\boldsymbol{w}$ is the weight vector,

$$\boldsymbol{u} = [u_1\ u_2\ \ldots\ u_m]^T$$

$$\boldsymbol{w} = [w_1\ w_2\ \ldots\ w_m]^T$$

GADALINE structure is shown in Fig. 17.

3.10.2 GADALINE Learning Algorithm

The GADALINE learning algorithm is based on the Widrow-Hoff rule with two types of generalizations to speed up convergence. The learning algorithm is derived in the following and the implication on the generalized learning will also be discussed. Widrow-Hoff's delta rule or Least Mean Square algorithm (LMS), is based on minimizing the cost function,

$$E(\boldsymbol{w}) = 1/2\ e^2(t)$$

where t is the iteration index and $e(t)$ is given by,

$$e(t) = \mathrm{d}(t) - y(t)$$

where the desired output $\mathrm{d}(t)$ at each iteration t is constant.

According to LMS, the weight adjustment is

$$\Delta \boldsymbol{w}(t) = \boldsymbol{w}(t+1) - \boldsymbol{w}(t) = -\eta\, e(t)\boldsymbol{u}(t) \tag{3.5}$$

here the learning–rate is usually in the range of $0 < \eta < 2$.

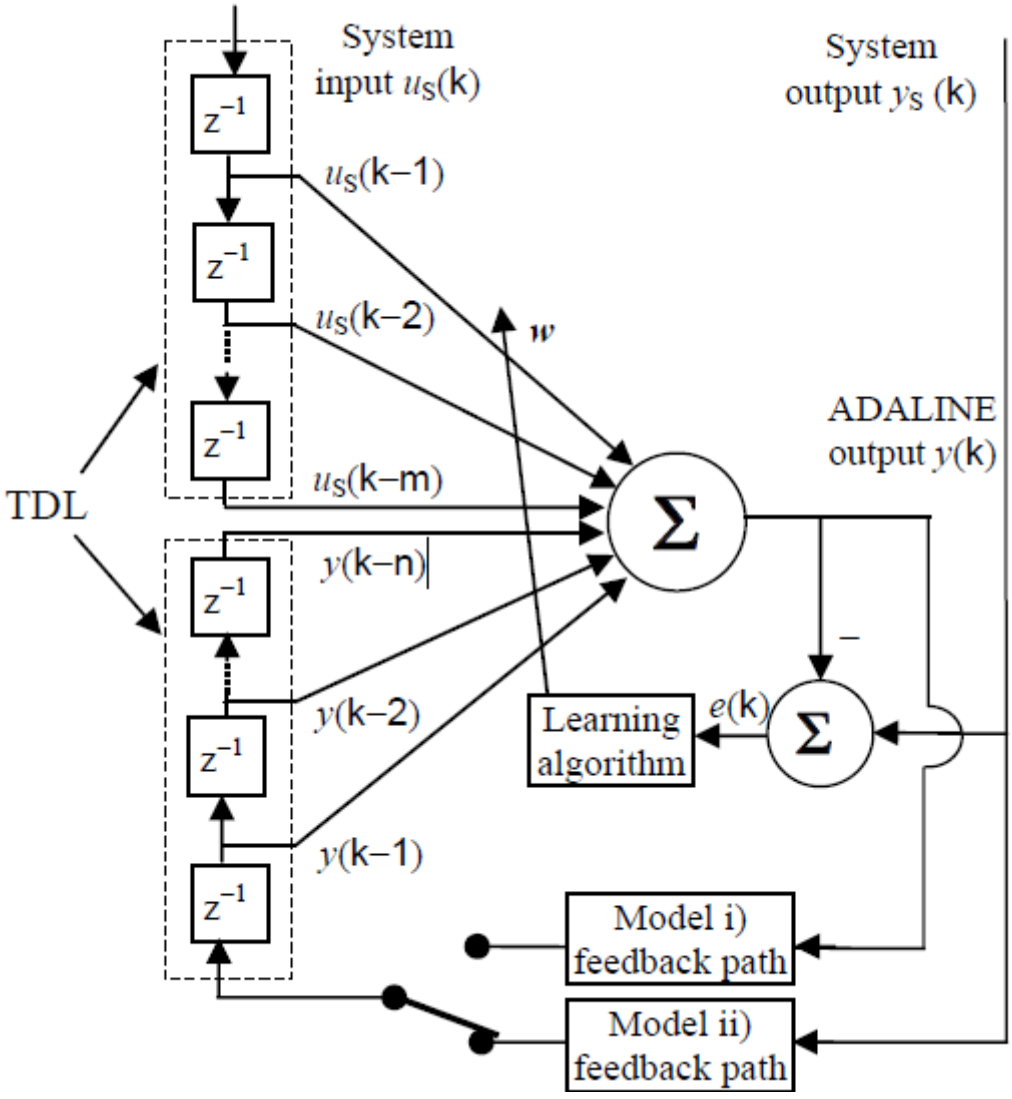

Figure 17 *GADALINE configuration [39]*

It is well known (Haykin 1999), the LMS algorithm has a slow convergence speed. Especially for bigger learning parameter, the convergence trajectory exhibits a zigzag form. This can be stabilized partially by normalizing input vector to a unit vector in the weight adjustment equation (3.5) so that the weight adjustment magnitude is independent of the magnitude of the input vector. That is, the weight adjustment is given by,

$$\Delta \boldsymbol{w} = \eta\, e\, \boldsymbol{u}/\|\boldsymbol{u}\|$$

Where $\|\boldsymbol{u}\|$ is the Euclidean norm of the input vector $\boldsymbol{u}$ or the length, which is calculated as,

$$\|\boldsymbol{u}\| = u_1^2 + u_2^2 + \ldots + u_m^2$$

In order to speed up the convergence and thus to increase the capability of tracking time varying system parameters, we propose two techniques: i) introducing a momentum term into the weight adjustment equation (3.5); and ii) training multiple epochs with data from a sliding window of the entire set of input samples.

3.10.2.1 Momentum in Learning

With a momentum term introduced into the weight adjustment equation (3.5), the new weight adjustment now becomes,

$$\Delta w(t) = \eta\, e(t)\, \boldsymbol{u}(t) + \alpha\, \Delta \boldsymbol{w}(t-1) \qquad (3.6)$$

Where $1 > \alpha >= 0$ is a small non-negative number.

To see the effect of the momentum, we can expand the equation starting from $t = 0$,

$$\Delta \boldsymbol{w}(0) = \eta\, e(0)\boldsymbol{u}(0)$$

$$\Delta \boldsymbol{w}(1) = \eta\, e(1)\boldsymbol{u}(1) + \alpha\Delta \boldsymbol{w}(0) = \eta\, e(1)\boldsymbol{u}(1) + \alpha\, \eta\, e(0)u(0)$$

$$\Delta \boldsymbol{w}(2) = \eta\, e(2)\boldsymbol{u}(2) + \alpha\, \Delta \boldsymbol{w}(1) = \eta\, e(2)\boldsymbol{u}(2) + \alpha\, \eta\, e(1)\boldsymbol{u}(1) + \alpha^2\, \eta\, e(0)\boldsymbol{u}(0)$$

$$\ldots$$

$$\Delta \boldsymbol{w}(t) = \eta\, e(t)\, \boldsymbol{u}(t) + \alpha\, \Delta \boldsymbol{w}(t-1) = \eta\, e(t)\boldsymbol{u}(t) + \alpha\, \eta\, e(t-1)\boldsymbol{u}(t-1) + \ldots + \alpha^{n-1}\, \eta\, e(1)\boldsymbol{u}(1) + \alpha^n\, \eta e(0)\boldsymbol{u}(0)$$

$$= \eta \sum \alpha^{t-\tau}\, e(\tau)\, \boldsymbol{u}(\tau)$$

Remember $e(t)\boldsymbol{u}(t)$ is just $-\partial E(t)/\partial \boldsymbol{w}(t)$, then

$$\Delta \boldsymbol{w}(t) = -\eta \sum \alpha^{t-\tau} \partial E(t)/\partial \boldsymbol{w}(t)$$

So, we have an exponentially weighted time series. We can then make the following observations,

1) If the derivative $\partial E(t)/\partial \boldsymbol{w}(t)$ has opposite sign on consecutive iterations, the exponentially weighted sum $\Delta \boldsymbol{w}(t)$ reduces in magnitude, so the weight $\boldsymbol{w}(t+1)$ is adjusted by a small amount, which reduces the zigzag effect in weight adjustment.
2) On the other hand, if the derivative $\partial E(t)/\partial \boldsymbol{w}(t)$ has same sign on consecutive iterations, the weighted sum $\Delta w(t)$grows in magnitude, so the weight $w(t+1)$ is adjusted by a large amount. The effect of the momentum term is to accelerate descent in steady down hill region. Because of this added momentum term, the weight adjustment continues after convergence, that is, when even the magnitude of the error term *e* is small. For linear system identification purpose, these weights correspond to system parameters. It is desirable to keep these parameters fixed once convergence is seen. So we should turn off the momentum term after convergence is detected, say $|e| < \varepsilon$, $\varepsilon > 0$ is a small number. However, when the system parameters change at a later time, another convergence period will begin. Then it is better to turn the momentum back on.

GADALINE training is performed by presenting the training pattern set – a sequence of input-output pairs. During training, each input pattern from the training set is presented to the GADALINE and it computes its output *y*. This value and the target output from the training set are used to generate the error and the error is then used to adjust all the weights.

3.10.2.2 Multiple Epochs Training with a Sliding Window

Another technique to increase the convergence speed and the capability of tracking time varying system parameters is to train the ADALINE network multiple epochs with data obtained from a sliding window of the entire set of input samples. Assume the size of the sliding window is *s*.

Then only the most recent s data samples will be used to adapt the weights of the ADALINE. At iteration t, the data set for an epoch is,

$$\{\boldsymbol{u}(t\text{-}s\text{+}1), \ldots, \boldsymbol{u}(t)\}$$

And the data set for iteration t +1 becomes,

$$\{\boldsymbol{u}(t\text{-}s\text{+}2), \ldots, \boldsymbol{u}(t\text{+}1)\}$$

Since the old data are discarded as t increases, this technique intuitively can follow well with the time changing parameters. And training multiple epochs on each sliding data set will further enhance this tracking capability. When the size of the sliding window is set to one, s =1, the training method is equivalent to normal LMS. Normally, this size is set to the number of inputs to the ADALINE. A bigger sliding window, however, will tend to stabilize the learning process. The training procedure based on this technique, called SWLMS. In the SWLMS algorithm, the size of the sliding window and the number of epochs on each data set are two tuning parameters. The larger the sliding window the slower is the convergence and the larger the number of epochs the faster the convergence and thus better parameter tracking capability. Large sliding window and increased number of epochs also increase the computational efforts in the multiples of sn_e, though today's computer is more powerful than enough.

The Epoch Interpretation of Learning

In other words, the ultimate goal of the neural network researcher is to build networks that provide optimal generalization performance. Once a network is trained, we wish it to perform well on examples that are not included in the training set. "Epoch interpretation of learning" can be used to improve the performance of early stopping based techniques used for improving generalization performance in neural networks. Experiments performed on noisy, non-linear foreign exchange rate data demonstrate that networks built using an early-stopping technique that uses the epoch interpretation of learning on average out-perform networks built using a conventional early-stopping technique by 11% [56].

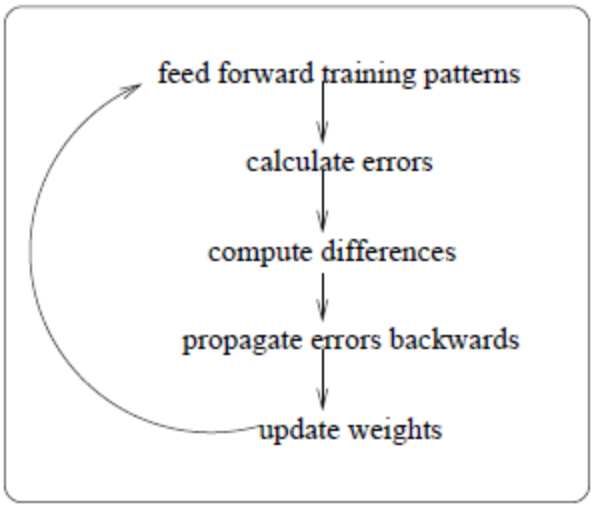

Figure 18 *The back-propagation neural network epoch [56]*

The epoch in back-propagation learning is one weight update or training iteration (see figure 18). For each epoch, the back-propagation learning algorithm builds a different model (i.e.) a network with a different set of weights. If a neural network is trained to 1000 epochs, the learning algorithm investigates or moves through 1000 different models. Neural network learning then, can be viewed as a search through a large number of models, for a model that has the set of weights that will provide the best generalization performance. The number of hidden units a network has also influences the generalization performance of a neural network and can be easily incorporated into this view of learning [57]. This performance improvement is most noticeable when noisy non-linear training data is used.

CHAPTER 4

PROBLEMS HANDLING USING GADALINE WITH DIFFERENT TECHNIQUES

4.1 System Identification

System identification is to determine the model structure and parameters for a dynamic system based on the measurable input and output data of the system. Consider a discrete time linear SISO system, the observable input and output data can be given in the form of a time series: $\{u(k\mathrm{T}), y(k\mathrm{T})\}$, T is the sample period. The system can be modeled by the following difference equation,

$$y(k) + a_1 y(k-1) + a_2 y(k-2) + \ldots + a_n y(k-n) = b_1 u(k-1) + b_2 u(k-1) + \ldots + b_m u(k-m) \quad (4.1)$$

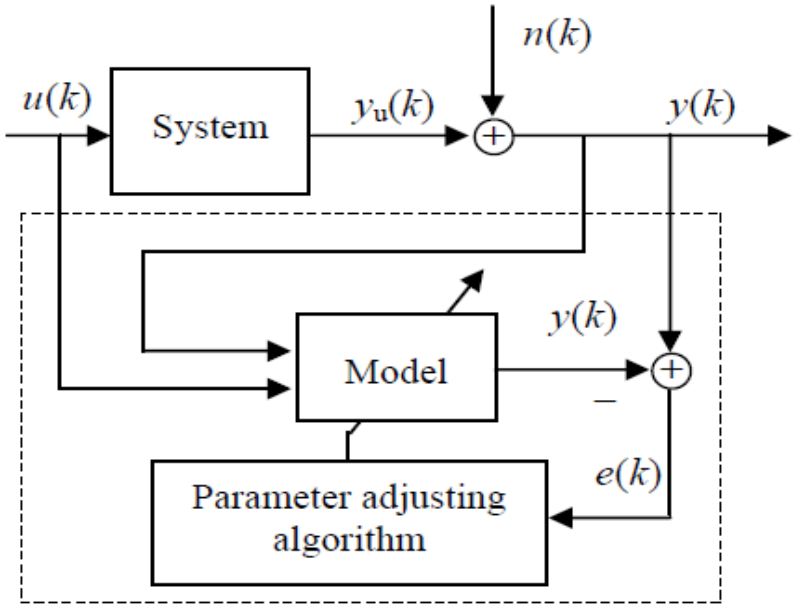

Figure 19 *System Identification [30]*

Where n and m are system structure parameters, $m < n$, n is the order of the system; a_i and b_j are system parameters, $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m$. In the time varying case, a_i and b_j can be slowly changing with time or can be switched to different values at different time instants. Then, the system identification becomes a parametric estimation problem that is, given n and m, to determine coefficients a_i and b_j from the input and output data of the system according to a

certain criterion, such as minimizing an error function, which is a measure of how close the model is to the actual system. Figure 19 shows a general system identification architecture, $y_u(k)$ is the deterministic output, $n(k)$ is the noise (white) and $e(k)$ is the error. The minimization can be achieved by adjusting the filter coefficients or weights in the neural network. The error signal to be used depends on the application. The criteria used for the minimization of the mean square error is the temporal average of the least squares error. Different algorithms are used for each of the minimization criteria e.g. the Least Mean Squares (LMS) algorithm, the Recursive Least Squares (RLS) algorithm etc. To understand the concept of adaptive noise cancellation, we use the minimum mean-square error criterion. Although the ADALINE network can only solve linearly separable problems it has been and is today one of the most widely used neural networks found in practical applications. Adaptive filtering is one of its major application areas.

4.2 Noise Cancellation

Noise isn't just irritating, it can also be costly. It can lead to mistakes in production because it can affect how machines work. That's why researchers are finding ways to reduce noise at its origin. Engines can be loud - loud enough to make a whole bus vibrate so much that the windows shake. Even small cars are noisy. The solution for some is to turn up the radio to drown out the engine. But there are better solutions than just trying to be louder than the noise. By producing similar but inverted sound waves, the disturbing noise can actually be canceled out. Three things are needed to produce suitable inverted waves, which cancel the noise: A sensor is required to detect the noise, a measurement technique to evaluate it, and an oscillator to produce the vibration for the inverted sound waves.

It is sound field modification, particularly sound field cancellation, by electro-acoustical means. In its simplest form, a control system drives a speaker to produce a sound field that is an exact mirror-image of the offending sound (the "disturbance"). The speaker thus "cancels" the disturbance, and the net result is no sound at all. In practice, of course, active control is somewhat more complicated. Panphonics concept of noise control and thin structure of transducers contribute in solving problems of converting and time-delay. This complicated audio problem becomes almost non-persistent with panphonics speakers. The name Active Noise

Cancellation differentiates "active" from traditional "passive" methods for controlling unwanted sound and vibration. Passive noise control treatments include insulation, silencers, vibration mounts, damping treatments, absorptive treatments such as ceiling tiles, and conventional mufflers like the ones used in today's automobiles. Passive techniques work best at middle and high frequencies, and are important to nearly all products in today's increasingly noise-sensitive world. However, passive treatments can be bulky and heavy when used for low-middle and low frequencies. However, passive treatments can be bulky and heavy when used for low frequencies. The size and mass of passive treatments usually depend on the acoustic wavelength, making them thicker and more massive for lower frequencies. Active noise control using panphonics elements as sensors and actuators makes it possible to change the acoustics of a room according to the special requirements at hand. Passive sound control does not always provide effective sound attenuation. Thick absorption layers and special structures are often required, resulting in higher costs and increased weight. Low frequencies in particular may require a large amount of passive sound absorption material, and in many cases, this is not a feasible solution.

There are many noise cancellation applications which require utilization of adaptive filters. An adaptive noise canceller adaptively filters a noise reference input to maximally match and subtract out noise or interference from the primary (signal plus noise) input. In this work we use a simple neural network called ADALINE as adaptive filter.

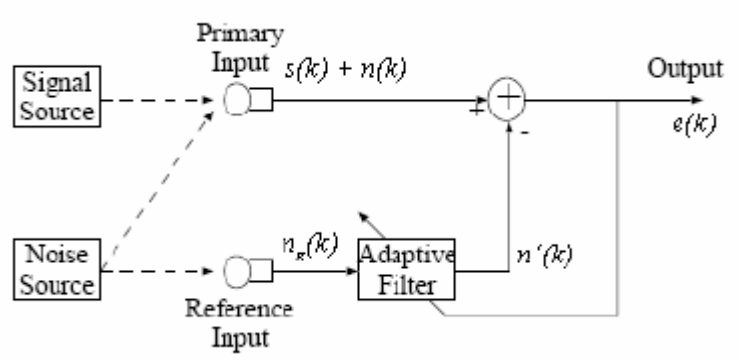

Figure 20 *Adaptive noise cancellation scheme [45]*

The primary input consists of speech signal *s(k)* and noise *n(k)* while the reference input consists of noise $n_R(k)$alone. Noise *n(k)* is not the same as noise $n_R(k)$, it is attenuated and filtered by

noise path. There is also a delay due to acoustic propagation. For maximal cancellation, an adaptive filter is thus needed to make $n_R(k)$ as similar as possible to the $n(k)$.

ADALINE is adaptively trained to minimize mean-square error $E[e^2(k)]$.

$$E\,[e^2\,(k)] = E\,[(s(k) + n(k) - n\,'(k))] \tag{4.2}$$

We assume $s(k)$, n(k) and $n_R(k)$ are zero-mean signals, and that $s(k)$ is independent of $n(k)$ and $n_R(k)$.Then

$$E\,[s(k)(n(k) - n'(k))] = 0$$

and

$$E\,[e^2\,(k)] = E\,[s^2\,(k)] + E\,[(\,n(k) - n\,'(k))^{\,2}\,]$$

Since the reference input signal has no information about $s(k)$ in it, minimizing $E[e^2(k)]$ can only affect the second term. In other words the minimization of $E[e^2(k)]$ is equivalent to the minimization of the difference between $n(k)$ and $n'(k)$. This will result in removing the engine noise from the contaminated signal, leaving the speech as the "error". The minimization can be achieved by adjusting the filter coefficients or weights in the neural network. The error signal to be used depends on the application. The criterion which is used for the minimization of the mean square error is the temporal average of the least squares error. Different algorithms are used for each of the minimization criteria e.g. the least mean squares (LMS) algorithm, the recursive least squares (RLS) algorithm etc. To understand the concept of adaptive noise cancellation, we use the minimum mean-square error criterion. Although the ADALINE network can only solve linearly separable problems it has been and is today one of the most widely used neural networks found in practical applications. Adaptive filtering is one of its major application areas.

In other words, in above figure 20, the method uses a “primary” input containing the corrupted Signal and a “reference” input containing noise correlated in some unknown way with the primary noise. The reference input is adaptively filtered and subtracted from the primary input to obtain the signal estimate. Adaptive filtering before subtraction allows the treatment of inputs that are deterministic or stochastic, stationary or time variable. And it is observed that when the reference input is free of signal and certain other conditions are met noise in the primary input can be essentialIy eliminated without signal distortion. Filters used for the above purpose can be

fixed or adaptive. The design of fixed filters is based on prior knowledge of both the signal and the noise. Adaptive filters, on the other hand, have the ability to adjust their own parameters automatically, and their design requires little or no apriori knowledge of signal or noise characteristics. Noise cancelling is a variation of optimal filtering that is highly advantageous in many applications. It makes use of an auxiliary or reference input derived from one or more sensors located at points in the noise field where the signal is weak or undetectable. This input is filtered and subtracted from a primary input containing both signal and noise. As a result the primary noise is attenuated or eliminated by cancellation [46].

At first glance, subtracting noise from a received signal would seem to be a dangerous procedure. If done improperly it could result in an increase in output noise power. If, however, filtering and subtraction are controlled by an appropriate adaptive process, noise reduction can be accomplished with little risk of distorting the signal or increasing the output noise level. In circumstances where adaptive noise cancelling is applicable, levels of noise rejection are often attainable that would be difficult or impossible to achieve by direct filtering.

There are a number of great applications for active noise cancellation devices. Noise cancellation almost requires the sound to be cancelled at a source, such as from a loud speaker. That is why the effect works well with headsets, since you can contain the original sound and the canceling sound in an area near your ear. One obvious application is that people working near aircraft or in noisy factories can now wear these electronic noise cancellation headsets to protect their hearing. ANC is ideal for industrial use. The application of active noise reduction produced by engines has various benefits -- the operation of the engines is more convenient for personnel; Noise reduction eliminates vibrations that cause material wearout and increased fuel consumption; Quieting of submarines. The concepts of noise cancellation have abundant applications; the cancelling of various forms of periodic interference in electrocardiography, the cancelling of periodic interference in speech signals, and the candling of broad-band interference in the sidelobes of an antenna array, also, a sine wave and Gaussian noise can be separated by using a reference input that is a delayed version of the primary input, these are just to name a few.

4.3 Adaptive Prediction

In prediction, a filter is used to estimate future values of a signal from prior observations. Figure 21 shows a closed-loop, adaptive predictor where d(n) is the desired signal, u(n) is the adaptive filter input, y(n) is the predicted signal, and e(n) is the prediction error. Although a single delay is shown, the delay could be as many samples as desired. If the predicted signal is the desired output, then this configuration is known as a prediction filter or predictor. If the error signal is the desired output, then the structure is called a prediction error filter.

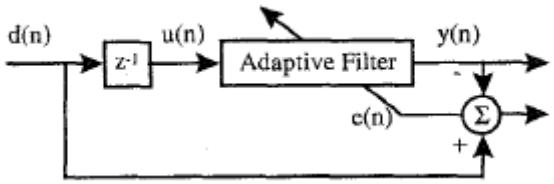

Figure 21*Single-step adaptive prediction filter[46]*

The performance of a predictor is measured using a quantity known as prediction gain [58]. The prediction gain is given by the ratio

$$G_p = \frac{\sigma_d^2}{\sigma_e^2}$$

Where σ_d^2 is the variance of the predictor input, d(n), and σ_e^2 is the variance of the prediction error, e(n). The minimization can be achieved by adjusting the filter coefficients or weights in the neural network. The error signal to be used depends on the application. The criteria used for the minimization of the mean square error is the temporal average of the least squares error. Different algorithms are used for each of the minimization criteria e.g. the Least Mean Squares (LMS) algorithm, the Recursive Least Squares (RLS) algorithm etc. To understand the concept of adaptive noise cancellation, we use the minimum mean-square error criterion. Although the ADALINE network can only solve linearly separable problems it has been and is today one of the most widely used neural networks found in practical applications. Adaptive filtering is one of its major application areas.

Thus, given the same input signal, the better predictor produces a smaller error variance yielding a larger prediction gain. The error variance may be estimated using the time average

$$\sigma_e^2 = \frac{1}{M} \sum_{n=1}^{M} e(n)^2$$

And the input variance may be estimated using

$$\sigma_d^2 = \frac{1}{M} \sum_{n=1}^{M} (d(n) - \bar{d})^2$$

Where $\bar{d}$ is the mean of the input signal.

The input signal of an adaptive filter operating using a linear prediction configuration is a delayed version of the signal to be predicted, d(t-T), and then the desired signal is given by d(t). The usage of ADALINE as a predictor in model predictive control is possible for both cases, despite the predictor was trained for data without noise. Due to influence of noise, the parameter ρ has to be increased to reduce the jittering of control actions [58].

The simulations results have proved that simple one-neuron network with linear transfer function is able to predict the nonlinear system output with moderate deviations. Moreover, it was shown that the ADALINE can be used for sufficient predictive control of this kind of systems. The main advantages of ADALINE are small memory requirements, fast training and simple usage. As a result of fast training time can be ADALINE easily adapted on-line, what increases the accuracy of control.

4.4 Techniques/ methods:

Learning Algorithms

Learning algorithm is based on the Widrow-Hoff rule.

Widrow-Hoff's delta rule or Least Mean Square algorithm (LMS), is based on minimizing the cost function, $E(\boldsymbol{w}) = 1/2\ e^2(t)$ where t is the iteration index and $e(t)$ is given by,

$$e(t) = \mathrm{d}(t) - y(t)$$

Where the desired output $\mathrm{d}(t)$ at each iteration is constant.

According to LMS, the weight adjustment is

$$\Delta \boldsymbol{w}(t) = \boldsymbol{w}(t+1) - \boldsymbol{w}(t) = -\eta * e(t) * \boldsymbol{u}(t),\ 0 < \eta < 2. \tag{4.3}$$

In order to speed up the convergence and thus to increase the capability of tracking time varying system parameters, we propose two techniques: i) introducing a momentum term into the weight adjustment equ. (4.2); and ii) training multiple epochs with data from a sliding window of the entire set of input samples.

The new weight adjustment now becomes,

$$\Delta \boldsymbol{w}(t) = -\eta_k * e(t) * \mathbf{u}(t) + \alpha * \Delta \boldsymbol{w}(t-1) \tag{4.4}$$

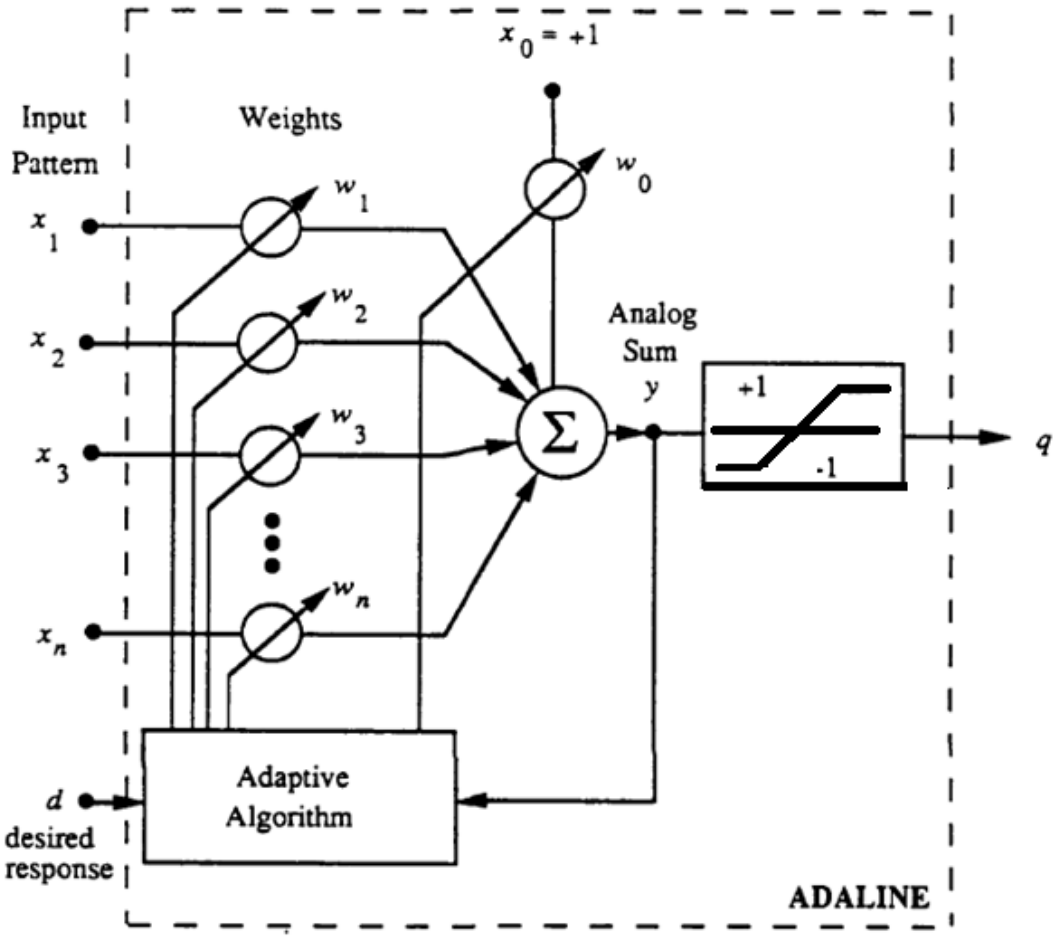

Figure 22 *The ADALINE [54]*

Here, we propose a new algorithms which are referred to as the variable learning-rate algorithms,

4.4.1 Method 1: A Variable Learning-rate Parameter LMS Algorithm [49]:

A new adaptive filter with a variable learning-rate parameter is introduced. The learning-rate parameter increases or decreases as the mean-square error increases or decreases, allowing the adaptive filter to track changes in the system as well as produce a small steady state error.
for adjusting the variable learning-rate parameter η_k:

$$\eta_{k+1} = \alpha * \eta_k + \gamma * e^2(t) \quad (4.5)$$

With

$0 < \alpha < 1, \quad \gamma > 0$

The learning-rate parameter is always positive and is controlled by the size of the prediction error and the parameters α and γ.

And

$$\begin{aligned} \eta_{k+1} &= \eta_{max} && \text{if } \eta_{k+1} > \eta_{max} \\ &= \eta_{min} && \text{if } \eta_{k+1} < \eta_{min} \\ &= 0 && \text{otherwise} \end{aligned} \quad (4.6)$$

where $\eta_{min} < \eta < \eta_{max}$.

The constant η_{max} is chosen to ensure that the mean-square error (MSE) of the algorithm remains bounded. η_{min} is chosen to provide a minimum level of tracking ability. Usually, η_{min} will be near the value of η that would be chosen for the fixed learning-rate parameter algorithm. A new LMS type algorithm has been introduced which uses a variable learning-rate parameter to reduce the tradeoff between misadjustment and tracking ability of the fixed learning-rate parameter algorithm. The variable learning-rate parameter algorithm also reduces sensitivity of the misadjustment to the level of nonstationarity A significant feature of this algorithm is that approximate formulas can be derived to predict the misadjustment in both stationary and nonstationary environments.

Comparison of the new algorithm with the fixed learning-rate parameter algorithm and another learning-rate parameter algorithm shows that it has superior performance which is already proven in the previous work.

4.4.2 Method 2: A Stochastic Gradient Adaptive Filter with Gradient Adaptive Learning-rate Parameter [50]:

The learning-rate parameter of the adaptive filter is changed according to a gradient descent algorithm designed to reduce the squared estimation error during each iteration. An approximate analysis of the performance of the adaptive filter is observed when its inputs are zero mean, white, and Gaussian for adjusting the variable learning-rate parameter η_k:

$$\eta_{k+1} = \eta_k + \rho * e(n) * e(n-1) * x_k(n) * x_k(n-1) \quad (4.7)$$

k = 0, 1, 2, ……. N-1

where ρ is a small positive constant.

This method has very good convergence speed and low steady-state misadjustment. Furthermore, the tracking performance of this algorithm in nonstationary environments is relatively insensitive to the choice of the parameters of the adaptive filter and is very close to the best possible performance of the least mean square (LMS) algorithm for a large range of values of the of the learning-rate parameter adaptation algorithm. The idea of the method is to change the time-varying convergence parameters in such a way that the change is proportional to the negative of the gradient of the squared estimation error with respect to the convergence parameter. Analysis shows that the steady-state misadjustment of other algorithms with variable learning-rate parameter depends on the initial value of learning-rate parameter. But this algorithm has very good convergence speeds as well as small misadjustments, irrespective of the initial value of learning-rate parameter. The performance of this algorithm is relatively independent of the choice of ρ in many nonstationary environments.

4.4.3 Method 3: A Robust Variable Learning-rate Parameter LMS-Type Algorithm [51]:

This algorithm has been the focus of much study due to its simplicity and robustness, leading to its implementation in many applications. Other algorithms have advantageous performance over the fixed learning-rate parameter generally only in a high signal-to-noise environment as they depend on instantaneous error that is contaminated by the disturbance noise. But the usefulness of any adaptive algorithm is judged by its performance in the presence of this noise. So their performances deteriorate in the presence of measurement noise. We then propose a new variable learning-rate parameter algorithm, where the learning-rate parameter of the algorithm is adjusted according to the square of the time-averaged estimate of the autocorrelation e(n) of e(n-1) and As a result, the algorithm can effectively adjust the learning-rate parameter while maintaining the immunity against independent noise disturbance.
for adjusting the variable learning-rate parameter η_k:

$$\eta_{k+1} = \alpha * \eta_k + \gamma * p^2(n) \tag{4.8}$$

with $0 < \alpha < 1$, $\gamma > 0$
and

$$p(n) = \beta * p(n-1) + (1-\beta) * e(n) * e(n-1) \tag{4.9}$$

Where $0 > \beta > 1$

The parameter γ controls the convergence time as well as the level of misadjustment of the algorithm. The use of p(n) in the update η_k of serves two objectives. First, the error autocorrelation is generally a good measure of the proximity to the optimum. Second, it rejects the effect of the uncorrelated noise sequence on the learning-rate parameter update. In the early stages of adaptation, the error autocorrelation estimate $p^2(n)$ is large, resulting in a large η_k . As we approach the optimum, the error autocorrelation approaches zero, resulting in a smaller learning-rate parameter. This provides the fast convergence due to large initial η_k while ensuring low misadjustment near optimum due to the small final η_k even in the presence of noise.

CHAPTER 5

SIMULATION RESULTS AND PARAMETERS

This chapter is divided into three parts: system identification, noise cancellation and adaptive prediction. The GADALINE neural configuration for these three problems are developed using Matrix Laboratory v-9.0.0. Computer simulation results are obtained using appropriate values of necessary parameters.

Comparison of results:

5.1 System Identification:

For purpose of simulation of [30] using three techniques (method1 [49], method2 [50], method3 [51]) input signal x(t) as shown in fig.23 is generated which is angle modulated signal with frequency 0.8 hertz. The total time taken in the problem is 6 sec. Total 1200 samples are taken from input signal using sampling frequency 200 hertz. This input signal is passed through a function with unknown system parameters whose dimensionality is three and output is generated which is also the desired or target signal. Same input is given to neural network, here, input matrix is generated of same size as dimensionality of given system and length of input samples. The predicted output signal generated from the neural network.

Then, comparison is made on different techniques/methods in calculating the error signal. Error signal is calculated by taking the difference between the desired and predicted output signal. Initial values to weights are randomly generated. Using the error signal, the weights of neural system get updated at each iteration till the error becomes minimum and hence, neural network is trained.

The weight updation is:

$$\Delta \boldsymbol{w}(t) = \boldsymbol{w}(t+1) - \boldsymbol{w}(t) = -\eta * e(t) * \mathrm{u}(t) \qquad (5.1)$$

Next, in order to speed up the convergence (or tracking), momentum term is added to weight adjustment. So the weight adjustment becomes

$$\Delta \boldsymbol{w}(t) = -\eta * e(t) * \boldsymbol{u}(t) + \alpha * \Delta \boldsymbol{w}(t-1) \qquad (5.2)$$

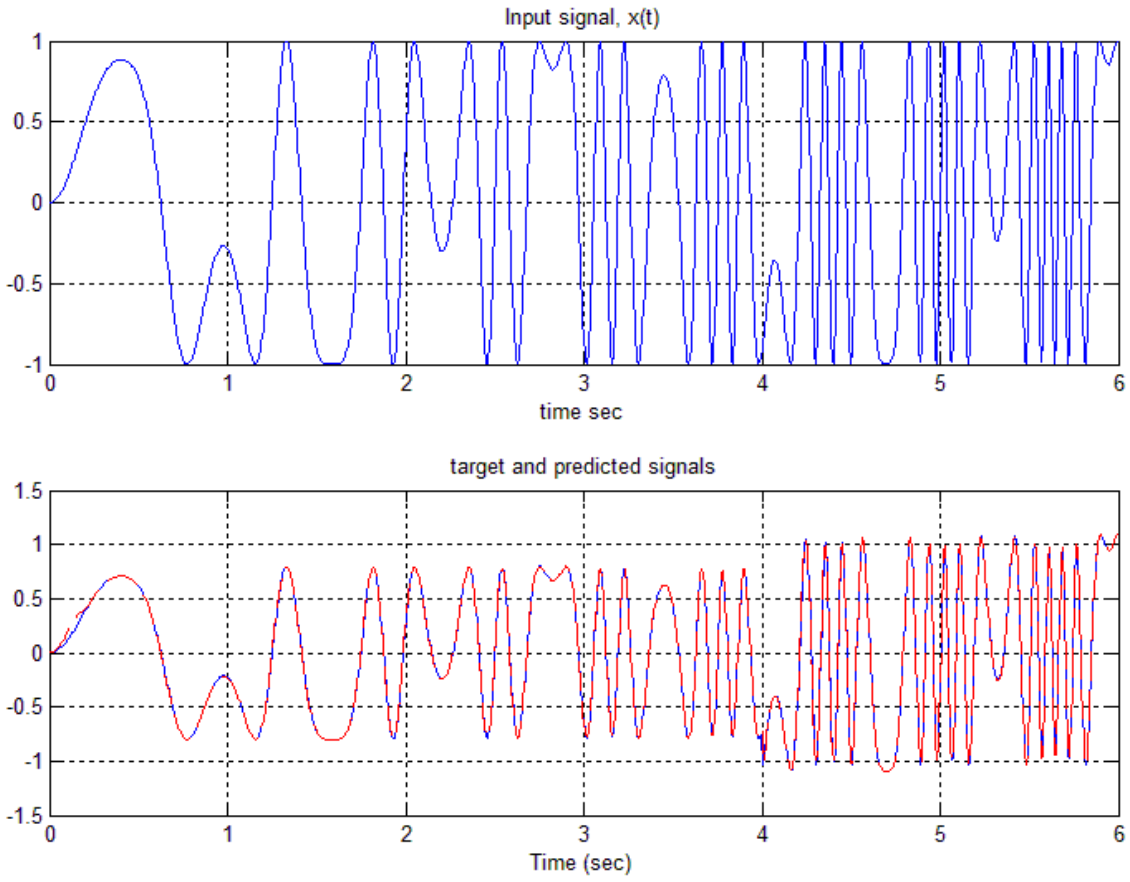

Figure 23 *Input and Target signal for System Identification*

Now, it is proposed that instead of constant learning-rate parameter (η), variable learning-rate parameter (η_k) is used, which is dependent on square of prediction error signal and above simulation parameters.

Comparison of weight adjustment of the three algorithms is shown in fig 24, it is observed that performance of method3 [51] is the best in terms of the error and performance of method2 [50] is not good and performance of method1 [49] lies between the other two.

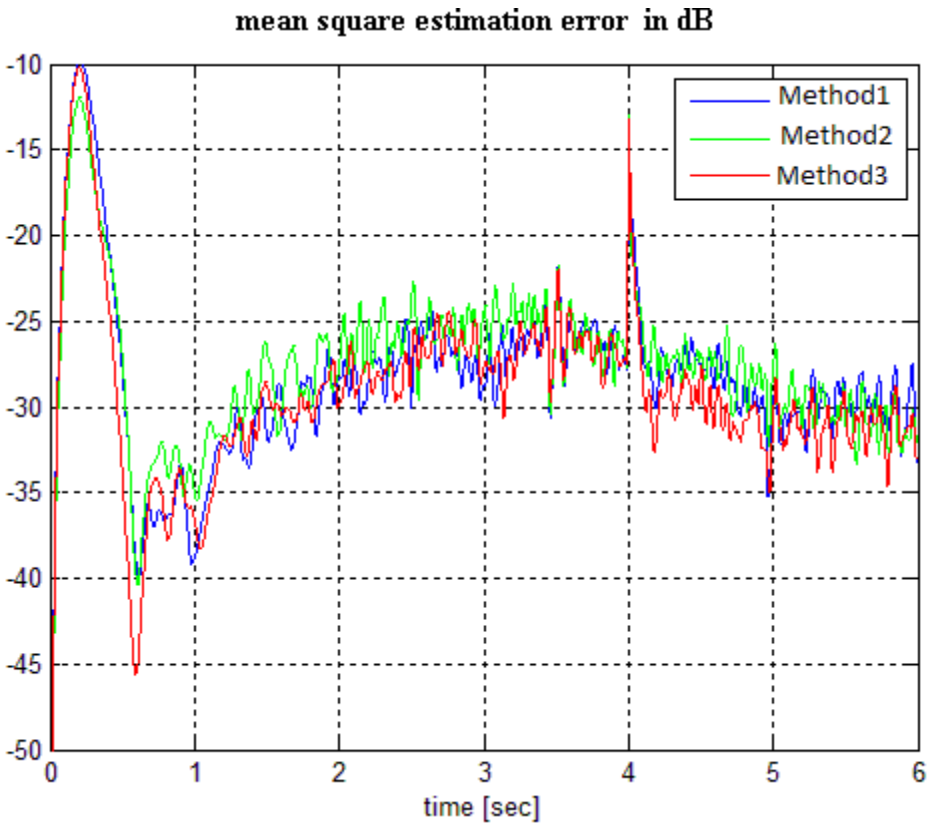

Figure 24 *Performance Comparison for System Identification*

The comparison is made keeping values of parameters same in all cases as:

Dimensionality = 3, $\eta = 0.2$;

and with other parameters as:

$\alpha = 0.99$, $\gamma = 0.01$ in equation

$$\eta_{k+1} = \alpha * \eta_k + \gamma * e^2(t) \qquad (5.3)$$

$\eta_{max} = 0.2$, $\eta_{min} = 0.18$ in equation

$$\eta_{k+1} = \eta_{max} \quad \text{if } \eta_{k+1} > \eta_{max}$$
$$= \eta_{min} \quad \text{if } \eta_{k+1} < \eta_{min}$$
$$= 0 \quad \text{otherwise} \tag{5.4}$$

$\rho = 0.01$ in the equation

$$\eta_{k+1} = \eta_k + \rho * e(n) * e(n-1) * x_k(n) * x_k(n-1) \tag{5.5}$$

$\alpha = 0.97$, $\gamma = 0.001$ in the equation

$$\eta_{k+1} = \alpha * \eta_k + \gamma * p(n)^2 \tag{5.6}$$

$\beta = 0.99$ in the equation

$$p(n) = \beta * p(n-1) + (1-\beta) * e(n) * e(n-1) \tag{5.7}$$

5.2 Noise Cancellation

For purpose of simulation of [45] using three techniques (method1 [49], method2 [50], method3 [51]) input signal u(t) as shown in fig 25 is generated which is frequency and amplitude modulated signal with frequency 4e3 hertz. The total time taken in the problem is 10 sec. Total 400 samples are taken from input signal using sampling frequency 1/2e-5 hertz. This input signal is passed through a function with unknown system parameters whose dimensionality is three and output is generated which is also the desired or target signal.

Same input is given to neural network, here; input matrix is generated of same size as dimensionality of given system and length of input samples. The predicted output signal generated from the neural network.

Then, comparison is made on different techniques/methods in calculating the error signal. Error signal is calculated by taking the difference between the desired and predicted output signal. Initial values to weights are randomly generated. Using the error signal, the weights of neural

system get updated at each iteration till the error becomes minimum and hence, neural network is trained.

The weight updation is:

$$\Delta \boldsymbol{w}(t) = \boldsymbol{w}(t+1) - \boldsymbol{w}(t) = -\eta * e(t*)\mathrm{u}(t) \tag{5.8}$$

Next, in order to speed up the convergence (or tracking), momentum term is added to weight adjustment. So the weight adjustment becomes

$$\Delta \boldsymbol{w}(t) = -\eta * e(t) * \boldsymbol{u}(t) + \alpha * \Delta \boldsymbol{w}(t-1) \tag{5.9}$$

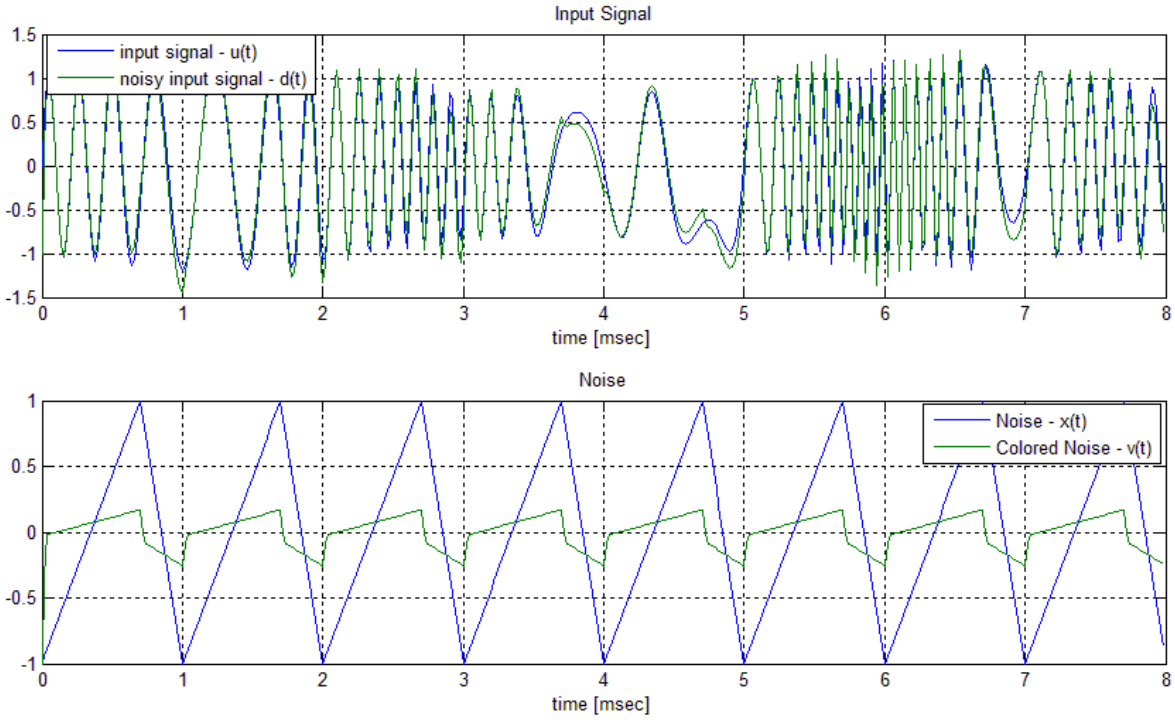

Figure 25 *Input and Noise Signal for Noise Cancellation*

Now, it is proposed that instead of constant learning-rate parameter (η), variable learning-rate parameter (η_k) is used, which is dependent on square of prediction error signal and above simulation parameters.

Comparison of weight adjustment of three algorithms is shown in fig 26, it is observed that performance of method3 [50] is the best in terms of the error and performance of method2 [49] is not good and performance of method1 [48] lies between the other two.

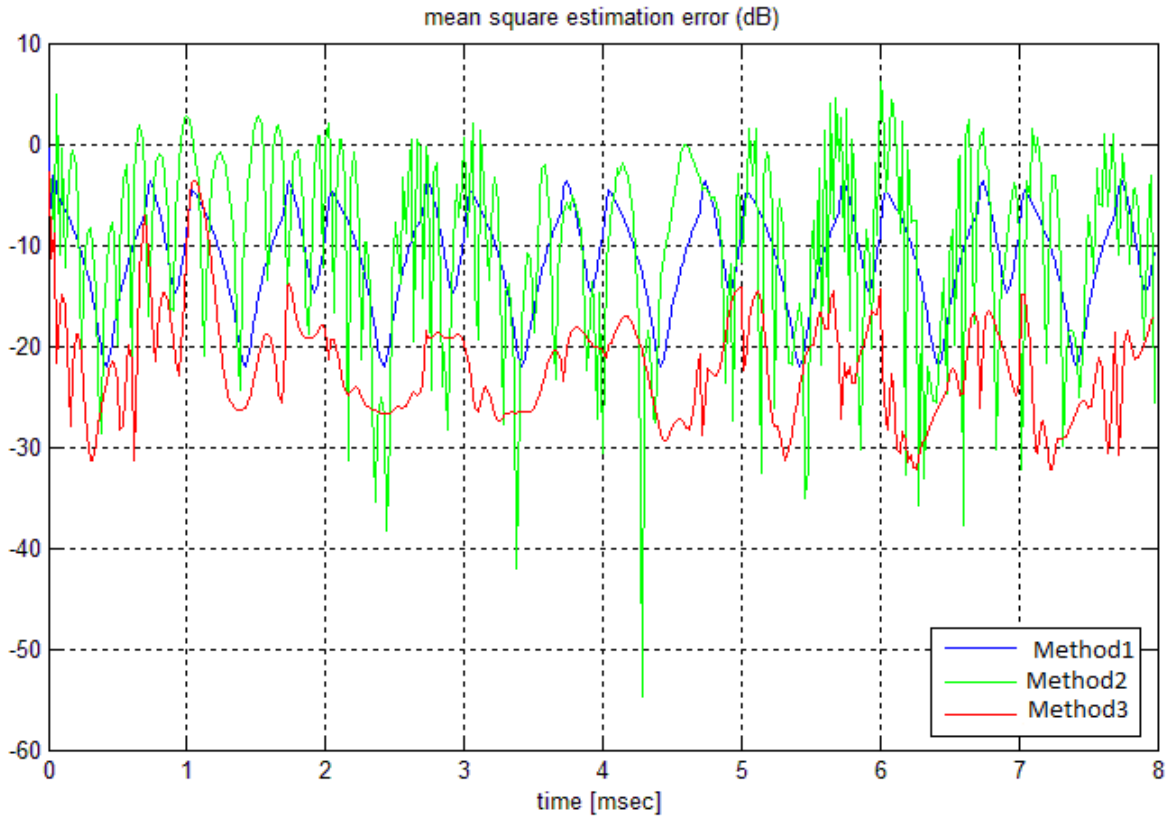

Figure 26*Performance Comparison for Noise Cancellation*

The comparison is made by keeping values of parameters same in all cases as:

Dimensionality = 4, $\eta = 0.05$;

and with other parameters as:

$\alpha = 0.95$, $\gamma = 0.01$ in equation

$$\eta_{k+1} = \alpha * \eta_k + \gamma * e^2(t) \qquad (5.10)$$

$\rho = 0.005$ in the equation

$$\eta_{k+1} = \eta_k + \rho * e(n) * e(n-1) * x_k(n) * x_k(n-1) \quad (5.11)$$

$\alpha = 0.95$, $\gamma = 0.01$ in the equation

$$\eta_{k+1} = \alpha * \eta_k + \gamma * p(n)^2 \quad (5.12)$$

$\beta = 0.99$ in the equation

$$p(n) = \beta * p(n-1) + (1-\beta) * e(n) * e(n-1) \quad (5.13)$$

5.3 Adaptive Prediction

For purpose of simulation of [48] using three techniques (method1 [49], method2 [50], method3 [51]) input signal x(t) as shown in fig 27 is generated which is angle modulated signal with frequency 2k hertz. The total time taken in the problem is 7.5 sec. Total 302 samples are taken from input signal using sampling frequency 80k hertz. This input signal is passed through a function with unknown system parameters whose dimensionality is three and output is generated which is also the desired or target signal.

Same input is given to neural network, here, input matrix is generated of same size as dimensionality of given system and length of input samples. The predicted output signal generated from the neural network.

Error signal is calculated by taking the difference between the desired and predicted output signal. Initial values to weights are randomly generated. Using the error signal, the weights of neural system get updated at each iteration till the error becomes minimum and hence, neural network is trained.

The weight updation is:

$$\Delta \boldsymbol{w}(t) = \boldsymbol{w}(t+1) - \boldsymbol{w}(t) = -\eta * e(t^*)u(t) \quad (5.14)$$

Next, in order to speed up the convergence (or tracking), momentum term is added to weight adjustment. So the weight adjustment becomes

$$\Delta \boldsymbol{w}(t) = -\eta * e(t) * \boldsymbol{u}(t) + \alpha * \Delta \boldsymbol{w}(t-1) \qquad (5.15)$$

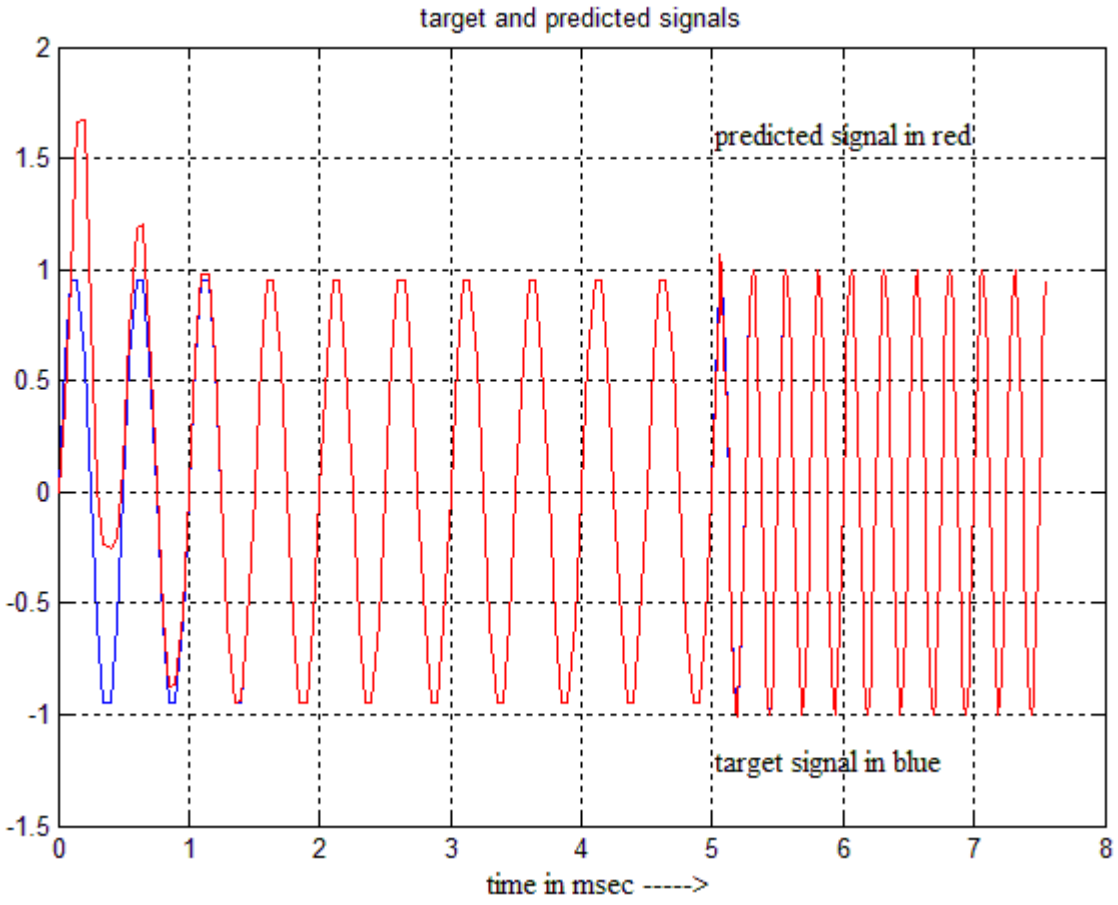

Figure 27 *Input and Target Signal for Adaptive Prediction*

Now, it is proposed that instead of constant learning-rate parameter (η), variable learning-rate parameter (η_k) is used, which is dependent on square of prediction error signal and above simulation parameters.

Comparison of weight adjustment of three algorithms is shown in fig 28, it is observed that performance of method3 [51] is the best in terms of the error and performance of method2 [50] is not good and performance of method1 [49] lies between the other two.

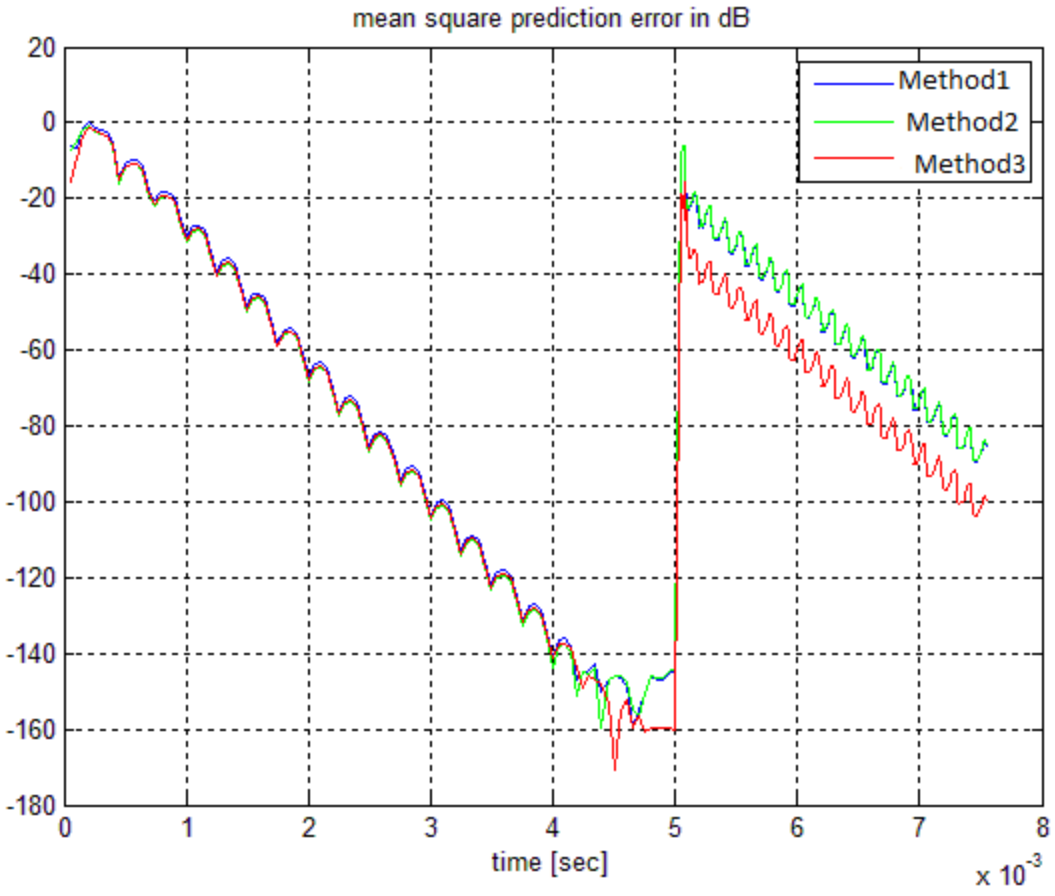

Figure 28 *Performance Comparison for Adaptive Prediction*

The comparison is made keeping values of parameters same in all cases as:

Dimensionality = 4, $\eta = 0.4$;

and with other parameters as:

$\alpha = 0.97$, $\gamma = 0.05$ in the equation

$$\eta_{k+1} = \alpha * \eta_k + \gamma * e^2(t) \qquad (5.16)$$

$\rho = 0.001$ in the equation

$$\eta_{k+1} = \eta_k + \rho * e(n) * e(n-1) * x_k(n) * x_k(n-1) \quad (5.17)$$

$\alpha = 0.97$, $\gamma = 0.001$ in the equation

$$\eta_{k+1} = \alpha * \eta_k + \gamma * p(n)^2 \quad (5.18)$$

$\beta = 0.99$ in the equation

$$p(n) = \beta * p(n-1) + (1-\beta) * e(n) * e(n-1) \quad (5.19)$$

CHAPTER 6
CONCLUDING REMARKS AND FUTURE SCOPE

GADALINE has long been studied and three different techniques have been implemented on it. These techniques use variable learning-rate parameter all in different manner. Previous work in this area has compared the results obtained with fixed learning-rate parameter algorithm and variable learning-rate parameter algorithm. The improvements in performance over the fixed learning-rate parameter are clearly shown. This algorithm is capable of giving both fast tracking as well as small misadjustment. In the case of fixed learning-rate constant, misadjustment level is achieved in case of small learning-rate constant and fast convergence rate is achieved using large learning-rate constant. So, the variable learning-rate parameter reduces the tradeoff between misadjustment and the speed of convergence.

The work in the thesis is an extension of the performance of algorithms using variable learning-rate parameter. We have further investigated the best algorithm with variable learning-rate parameter over other algorithms which also use variable learning-rate parameter. Simulation results show that the best algorithm (method) is having best performance and has a significant convergence rate improvement over those algorithms in a stationary environment for the same excess MSE in the same stationary environment. In this method the learning-rate parameter of the algorithm is adjusted according to the square of a time-averaging estimate of the autocorrelation of e(n) and e(n-1) instead of instantaneous error value e(n). Due to this reason, this algorithm gives best performance even in the presence of noise. Simulation results show that more flexible control of misadjustment and convergence time without the need to compromise one for the other.

A probabilistic approach for Generalized ADALINE neural configuration with variable learning-rate parameter is presented in this thesis. As our future work, we aim to present the modification in variable learning-rate parameter for improved convergence as well as tracking of Generalized ADALINE neural configuration for system identification and other problems.

REFERENCES

[1] K. Mehrotra, C. Mohan, S. Ranka, "*Elements of Artificial Neural Networks*," MIT press, 1997.

[2] S. Haykin, "*Neural Networks, A Comprehensive Foundation,"* 2nd edition, Prentice Hall, 1999.

[3] "*Artificial Neural Network,*" Wikipedia Encyclopedia, Wikimedia Foundation, Inc., 2013.

[4] B. Burke Harry, "Evaluating Artificial Neural Networks for Medical Applications," *IEEE Conference on Neural Networks,* Jun. 1997, pp. 2494-2495.

[5] W Hussain and W Ishak, "The Potential of Neural Networks in Medical Applications," *IEEE Conference on Neural Networks,* Jun. 2005, pp. 456-460.

[6] B. Pollack Jordan, "Connectionism: Past, Present and Future," *IEEE Proceeding*, vol.78, no. 9, pp. 1415-1442, Jun. 1998.

[7] L. Ljung, "*System Identification - Theory for the User,"* 2nd edition, Prentice-Hall, 1999.

[8] T. Söderström, and P. Stoica, "*System Identification,*" Prentice Hall, Eaglewood Cliffs, NJ. 1989.

[9] M. Atencia, and G. Sandoval, "Gray Box Identification with Hopfield Neural Networks," *Revista Investigacion Operacional,* vol. 25, no. 1, pp. 54-60, May 2004.

[10] J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," *IEEE International Conference on Neural Networks*, Jul. 2004, pp. 2555-2558.

[11] Ricardo Valverde, "Dynamic Systems Identification using RBF Neural Networks," *IEEE Transaction on Neural Networks,* vol. 5, pp. 3226-3230, Jul. 1999.

[12] A. Gretton, A. Doucet, R. Herbrich, P. Rayner and B. Schölkopf, "Support Vector Regression for Black-Box System Identification," *IEEE Transaction on Statistical Signal Processing*, vol. 3, pp. 341-344, Aug. 2001.

[13] J. Abonyi, and F. Szeifert, "System Identification using Delaunay Tessellation of Self-Organizing Maps," 6th *IEEE International Conference on Neural Networks and Soft Computing*, Jun. 2002, pp. 303-310.

[14] Landau, LD., "*Adaptive Control- The Model Reference Approach*," Marcel Dekker, New York, 1979.

[15] P. Kumar, "Theory and Practice of Recursive Identification*," IEEE Transaction on Automatic Control*, vol. 30, no. 10, pp. 1054-1056, Oct. 1995.

[16] P. Eykhoff, "*System Identification: Parameter and State Estimation,"* John Weley, New York, 1974.

[17] T. Soderstrom, P. Stoica, "*System Identification,"* Prentice Hall.NJ, 1989.

[18] K. *S.* Narendra and K. Parthasarathy, "Identification and Control of Dynamical Systems using Neural Networks," *IEEE Transaction on Neural Networks*, vol. 1, no. 1, pp. 4-27, Mar. 1990.

[19] K. *S.* Narendra and K. Parthasarathy, "Gradient Methods for Optimization of Dynamical Systems Containing Neural Networks," *IEEE Transaction on Neural Networks*, vol. 2, no. 2, pp. 252-262, Mar. 1991.

[20] R. Shoureshi, R. Chu, and M. Tenorio, "Neural Space Representation of Dynamic System," IEEE *Proceeding*, vol. 35, no. 4, pp. 457- 461, Apr. 1988.

[21] J. Hopfield and D. W. Tank, "Neural Computation of Decisions in Optimization Problems," *Biological Cybernetics*, vol. 52, no. 3, pp. 141-152, Jul. 1985.

[22] D. W. Tank and J. Hopfield, "Simple neural optimization networks: An A/D Converter, Signal Decision Circuit and Linear Programming Circuit," *IEEE Transaction on Circuits and Systems*, vol. 33, no. 5, pp. 533-541, May 1986.

[23] S. Bhama and M. H. Hassoun, “Continuous Hopfield Networks: Hardware Implementation,” *IEEE Transaction on Neural Networks*, vol. 69, no. 5, pp. 603-612, Nov. 1990.

[24] Ben Krose and Patrick van der Smagt, "*An Introduction to Neural Networks*," Prentice-Hall, Englewood Cliffs NJ., 1999.

[25] Simon Haykin, "*Adaptive Filter Theory*", Prentice Hall, 1996.

[26] B. Widrow et al., ‘‘Neural Networks for Adaptive Filtering and Adaptive Pattern Recognition,’’ *IEEE Computer*, vol. 21, no. 3, pp. 25-39, Mar. 1988.

[27] Shuitsu Matsumura, Takuji Maezaw*, ‘‘*Least-Square-Based Block Adaptive Prediction Approach for Lossless Image Coding,” *IEEE Conference on Circuit Theory and Design*, Aug. 2007, pp. 188-191.

[28] Y Tsukamoto, Akira Namatame, “Evolving Neural Network Models,” *IEEE International Conference on Evolutionary Computation*, May 1996, pp. 689-693.

[29] T. Lee, “*Structure Level Adaptation for Artificial Neural Networks,”* Kluwer Academic Publishers, 1991.

[30] Wenle Zhang, “System Identification Based on a Generalized ADALINE Neural Network,” *American Control Conference*, New York City, USA, Jul. 2007, pp. 4792-4797.

[31] J. Hopfield, and D.W. Tank, “Neural Computation of Decisions in Optimization Problems," *Biological Cybernetics*, vol. 52, May 1995.

[32] D.W. Tank, and J. Hopfield, "Simple Neural Optimization Networks: An A/D Converter, Signal Decision Circuit, and a Linear Programming Circuit," *IEEE Transaction on Circuits and System*, vol. 33, no. 12, pp. 533-541, May 2005.

[33] J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," *IEEE International Conference on Neural Networks*, Jul. 2004, pp. 2555-2558.

[34] J. Hopfield, , "Neurons with Graded Response Have Collective Computational Properties Like Those of Two-State Neurons," *IEEE International Conference on Neural Networks*, Jul. 2006, pp. 3088-3092.

[35] Reynold Chu and Manoel Tenorio, "Neural Networks for System Identification*," IEEE Control Systems Magazine*, pp. 31-34, Apr. 1990.

[36] Satyendra Bhama and Harpreet Singh, "Single Layer Neural Networks for Linear System Identification Using Gradient Descent Technique," *IEEE Transactions on Neural Networks*, vol. 4, no. 5, pp. 884-888, Sept. 1993.

[37] S. Bhama and H. Singh, "Dynamic image modeling by neural networks," *IEEE Transactions on Neural Networks,* vol. 25, no. 4, pp. 803-811, May 1995.

[38] *S.* Bhama and H. Singh, "Single Layer Neural Architecture for Dynamic Image Modeling," *IEEE Transactions on Neural Networks,* vol. 22, no. 3, pp. 1315-1322, Jul. 1998.

[39] Wenle Zhang, "On Training Optimization of the Generalized ADLINE Neural Network for Time Varying System Identification," *IEEE Chinese Control and Decision Conference (CCDC),* Jun. 2009, pp. 775-780.

[40] Wenle Zhang, "System Identification Based on an Improved Generalized ADALINE Neural Network," *IEEE Chinese Control and Decision Conference (CCDC), May* 2011, pp. 789-794.

[41] S.Z. Qin, H.T. Su and T.J. McAvoy, "Comparison of Four Neural Net Learning Methods for Dynamic System Identification," *IEEE Transaction on Neural Networks*, vol. 2, no. 1, pp. 52-262, Jan. 1992.

[42] Peter M. Mills and Albert Y. Zomaya, "A Neural Network Approach to On-line Identification of Non-linear Systems," *IEEE International Joint Conference on Neural Network,* Nov. 2006, pp. 202-207.

[43] N. K. Sinha. and B. Kuszta, "*Modelling and Identification of Dynamic Systems,"* Van Nostrand Reinhold *Co.*, New York, 1983.

[44] R. H. Middleton, G. C. Goodwin, "*Digital Control and Estimation,"* Prentice-Hall, Englewood Cliffs NJ., 1999.

[45] Maja Stella, "Adaptive Noise Cancellation Based on Neural Network,**"** *IEEE Computer*, vol. 21, no. 3, pp. 25-39, Mar. 1988.

[46] Bernard Widrow and John M. McCool, "Adaptive Noise Cancelling: Principles and Applications," *IEEE Transaction on Signal Processing,* vol. 63, no. 12, pp. 1692-1716, Dec. 1975.

[47] Shuitsu Matsumura, TakujiMaezaw*, "*Least-Square-Based Block Adaptive Prediction Approach for Lossless Image Coding," *IEEE Conference on Circuit theory and design*, Aug. 2007, pp. 188-191.

[48] David Samek, LubomirMacku, "Adaptive Linear Network in Model Predictive Control", *13th International Research/Expert Conference,* TMT, Hammamet, Tunisia, 2009, pp. 416-424.

[49] Raymond H. Kwong and Edward W. Johnston, "A Variable Step Size LSM Algorithm", *IEEE Transactions on Signal Processing,* vol. 1, pp. 423-426, Jul. 1992.

[50] V. John Mathews, "A Stochastic Gradient Adaptive Filter with Gradient Adaptive Step Size", *IEEE Transactions on Circuits and Systems*, vol. 1, pp. 2075-2083, Jun. 1993.

[51] Tyseer Aboulnasr, "A Robust Variable Step-Size LMS-Type Algorithm: Analysis and Simulations" *IEEE Transactions on Signal Processing,* vol. 45, no. 3, pp. 631-639, Mar. 1997.

[52] J. Randall, "An Introduction to Back Propagation Neural Networks", *Pharmaceutical Research,* vol. 10, no. 2, pp. 165-169, May 2004.

[53] Bernard Widrow and R. A. Baxter, "Layered Neural Networks for Pattern Recognition," *IEEE Transactions on Neural Networks,* vol. 36, pp. 1109-1118, Jun. 2001.

[54] B. Widrow, "An Adaptive "ADALINE" Neuron using Chemical ‘Memistors’," *IEEE Transaction on Neural Networks*, vol. 77, pp. 1553-1562, Oct. 1998.

[55] B. Widrow and C. Mays , "Learning with Critic Adaptive Threshold Systems," *IEEE Transactions on Systems and Cybernetics",* vol. 3, no. 5, pp. 455-465, Sep. 1993.

[56] John G. Carney and P´adraig Cunningham “The Epoch Interpretation of Learning,” *IEEE Transaction on Neural Networks*, vol. 8, pp. 111-116, Jun. 1998.

[57] J. Hertz, A. Krogh and R. Palmer, “*Introduction to the Theory of Neural Computation,"* Addison-Wesley Publishing Company, California, 1991.

[58] S. Jayant and P. Noll., “*Digital Coding* of *Waveforms,"* Eaglewood Cliffs, NJ: Prentice Hall, 1984