IMPLEMENTATION OF HEBBIAN-LMS ALGORITHM

A Thesis Submitted in Fulfillment of the Requirement for the Award of the Degree of

MASTER OF TECHNOLOGY

in VLSI DESIGN

Submitted By

VARTIKA

601562028

Under Supervision of

Ms. SAKSHI

ASSISTANT PROFESSOR

ELECTRONICS AND COMMUNICATION ENGINEERING DEPARTMENT

THAPAR UNIVERSITY, PATIALA, PUNJAB

JUNE, 2017

DECLARATION

I, **Vartika** hereby declare that the work presented in this thesis entitled "**Implementation of Hebbian-LMS Algorithm**" in fulfillment of the requirement for the award of degree of **Master of Technology(VLSI design)** submitted at ECED , Thapar University, Patiala is an authentic record of work carried out under supervision of **Ms. Sakshi** (Assistant Professor, ECED, Thapar University) from 2015 to 2017 The matter presented in this has not been submitted either in part or full to any other university or institute for the award of any other degree.

Vartika

Date: 12-09-2017

Vartika

Place: PATIALA

Roll No.601562028

CERTIFICATE

It is certified that the work contained in the thesis titled **IMPLEMENTATION OF HEBBIAN-LMS ALGORITHM** by **VARTIKA [601562028]** has been carried out under my supervision and that this work has not been submitted elsewhere for any other degree.

Sakshi

Ms. Sakshi
Assistant Professor
ECED
Thapar University, Patiala

Date: 12-09-2017

ACKNOWLEDGEMENT

It is my proud privilege to acknowledge and extend my gratitude to several persons who helped me directly or indirectly in completion of this report. I express my heart full indebtedness and owe a deep sense of gratitude to my teacher and my faculty guide **Ms. Sakshi, Assistant Professor** for their sincere guidance and support with encouragement to go ahead. I am also thankful to **Dr. Alpana Agarwal, Associate Professor and Head, ECED** for providing us with the adequate infrastructure for carrying out the work. I am also thankful to **Dr. Hemdutt Joshi**, **P.G. Coordinator, ECED and Dr. Anil Arora, Programme Coordinator, ECED** for the motivation and inspiration that

The study has indeed helped me to explore knowledge and avenues related to my topic and I am sure it will help me in my future.

Vartika

Roll No. 601562028

ABSTRACT

In most of the engineered network systems, a set of operations interact with each other in complex manners that can contain multiple types of relationships, depending on time and include other types of complexities. Such network systems comprise multiple subsystems and multiple layers of connectivity, and are generally called as multi-layer networks. There exist multi-layer neural networks which are feed forward artificial neural network model that maps sets of input data onto a set of appropriate outputs. it has multiple layers with each layer fully connected to the next one. An artificial multi layers neural network contains three layers broadly defined as input layer, hidden layer and output layer in the same order as written. These network systems can be trained using both supervised and unsupervised algorithms. The widely used supervised learning in these systems is LMS learning algorithm whereas the unsupervised learning used is Hebbian learning. A form of LMS algorithm can be established to achieve unsupervised learning. In this way LMS can be used to implement Hebbian learning. This implementation of combined algorithms is called as Hebbian-LMS learning algorithm. Combining the two algorithms creates a new unsupervised learning algorithm that has application in practical engineering problems.

In this thesis, an artificial neural network is considered, whose hidden layer weights are adjusted using Hebbian-LMS algorithm and the output layer is trained using the original supervised LMS algorithm and the results are recorded for various datasets using this approach as training algorithm in order to determine the feasibility of proposed algorithm in networks required to solve some practical engineering problems.

TABLE OF CONTENTS

Pre-Pages……i-xi

Declaration……ii

Certificate……iii

Acknowledgement……iv

Abstract……v

Table of Contents……vi

List of Figures……ix

List of Tables……xi

Chapter 1 Introduction……1-10

1.1 Neural Network's Biological Analogy……1

1.2 Neural network Classification on the Basis of Topology……2

1.3 Activation Functions……5

1.4 Types of Learning……6

1.5 Intelligence and Complexity……6

1.6 Benefits and Drawbacks of Multilayer Neural Network……7

1.7 Motivation.....8

1.7.1 Enormous Applications of Neural Networks.....8

1.7.2 Exploration of a New Algorithm.....9

1.7.3 Growth of Machine Learning and Neural Networks.....9

1.8 Organization of Thesis.....10

Chapter 2 Literature Survey.…….....……..…….11-15

2.1 Introducing Neural Network.....11

2.2 Learning.....12

Chapter 3 Learning.....16-28

3.1 Introduction.....16

3.2 Supervised Learning.....18

3.2.1 LMS and its Variants.....20

3.3 Unsupervised Learning.....22

3.3.1 Hebbian Learning Rule.....24

3.3.2 Hebbian-LMS Learning Algorithm.....26

3.4 Some Other Basic Types of Learning Rules.....27

3.4.1 Error Correction Rule.....27

3.4.2 Boltzmann Learning Rule.....28

3.4.3 Competitive Learning Rule.....28

Chapter 4 Structure and Methodology of Proposed Algorithm.....29-33

4.1 Basic Network Structure.....29

4.2 Hebbian-LMS Learning Rule.....30

4.3 LMS Learning Rule.....32

Chapter 5 Experimental Results and Analysis.....34-41

5.1 Proposed Algorithm Results.....34

Chapter 6 Conclusion and Future Prospects.....42

6.1 Concluding Remarks…….....42

6.2 Future Prospects.....42

References.....44-46

List Of Publications.....47

Originality Report

LIST OF FIGURES

Figure 1.1 An artificial neuron.....1

Figure 1.2 Neural Network Classification.....3

Figure 1.3 Feed-Forward Network.....3

Figure 1.4 Recurrent Network.....4

Figure 1.5 Different Types of Activation Functions.....5

Figure 3.1 Learning Rules of ANN.....17

Figure 3.2 Flow Chart for the Supervised Learning Rule.....20

Figure 3.3 Unsupervised Learning Rule Flow Chart.....23

Figure 4.1 Multilayer Neural Network.....29

Figure 4.2 A neuron trained with Hebbian-LMS learning.....31

Figure 4.3 A neuron trained with LMS algorithm.....32

Figure 5.1 Bar Graph of Iris and Breast Cancer Datasets.....36

Figure 5.2 Bar Graph of Gesture Phase and Heart Disease Datasets.....37

Figure 5.3 Box Plot of Iris Dataset MSE.....39

Figure 5.4 Box Plot of Breast Cancer Dataset MSE.....39

Figure 5.5 Box Plot of Gesture Phase Dataset MSE.....40

Figure 5.6 Box Plot of Heart Disease Dataset MSE.....40

LISTS OF TABLES

Table 5.1 Average Percentage Success Rate of Iris and Breast Cancer Dataset.....35

Table 5.2 Average Success Rate of Heart Disease and Gesture Phase Dataset.....35

Table 5.3 Comparison of MSE (average) of Iris and Breast Cancer Dataset.....37

Table 5.4 Comparison of MSE (average) of Heart Disease and Gesture Phase Dataset.....38

CHAPTER 1

INTRODUCTION

There is a rapid and massive growth of machine learning in electronics industry. Artificial neural networks are one approach to machine learning in which different algorithms are used to train the network to make it learn. Artificial neural networks are the biologically inspired networks which have several applications such as classification, pattern recognition, character recognition, image compression, stock market prediction, security etc.

1.1 NEURAL NETWORK'S BIOLOGICAL ANALOGY

A neuron is a living organism's or biological cell that process and transmits information. It consists a structure called as synapse which are specialized sites where neurons send and receive information from other cells. Synapse is a structure that permits a neuron or nerve cell to pass a signal to other neurons or nerve cells. Natural neurons receive signals through synapses situated on the dendrites or membrane of the neuron. When the signals received are strong enough (reach a certain threshold), the neuron is activated and transmits a signal though the axon. This threshold reached signal can be sent to other synapses, which in turn activate other neurons.

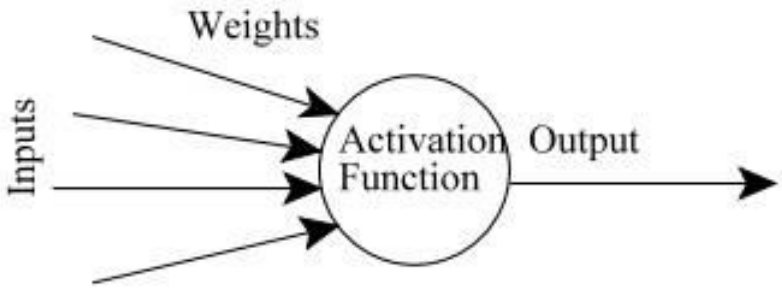


Figure 1.1 An Artificial Neuron

Figure 1.1 shows a structure of an artificial neuron inspired from biological neuron. Artificial neural networks are often presented as interconnected systems of "neurons" which exchange

information or data between each other. The connections have numeric weights that can be altered or adjusted, making neural networks more adaptive to inputs and equipped for learning. Artificial neural network consists inputs (like synapses), which are multiplied by weights (strength of the respective signals), and then calculated by a mathematical function which regulates the activation of the neuron. Another activation function computes the output.

The more greater a weight of an artificial neuron is, the more stronger the input which is multiplied by it will be. Basically a synaptic weight describes the connection between two neurons in case of artificial neural network. Depending on the weights, the calculation of the neuron will be different. By adjusting the weights of an artificial neuron, specified output could be obtained for respective input. When the artificial neuron network consists hundreds or thousands of neurons, it would be highly complicated to find weights manually. But there are number of algorithms which are used to adjust the neuron weights in order to achieve the desired or required output from the network. This process of altering and adjusting the neuron weights is called learning or training.

1.2 NEURAL NETWORK CLASSIFICATION ON THE BASIS OF TOPOLOGY

The feed-forward neural network and recurrent neural network can further be classified as shown in the figure 1.2. These networks are classified initially on the basis of connections as feed-forward and recurrent. Further these networks are classified on the basis of their structures. Single-layer perceptron, multilayer perceptron and radial basis functions nets are categorized under feed-forward neural networks whereas competitive, Khonen's SOM, Hopfield, ART models are grouped under the recurrent networks. Single layer and multilayer networks defines the number of hidden layers in the ANN. Radial basis function or RBF networks uses radial basis functions as their activation functions. The output obtained from the network is a linear combination of neuron parameters and input radial basis functions. Competitive network of recurrent networks is a form of unsupervised learning. For a set of inputs, nodes or neurons compete to respond for the right output. Kohonen is a type of self organizing NN, this property provides many new possibilities like adaptation to past unspecified input data. Hopfield networks processes the stationary set of inputs, and it has all symmetric connections. It can act as content addressable memory if trained by Hebbian learning which means they are resistant to connection modification. ART or adaptive

resonance theory models consists a different of neural networks which can be trained by using supervised as well as unsupervised learning and can be utilized in prediction and pattern recognition. In our research, single layer neural network is considered for the analysis purpose.

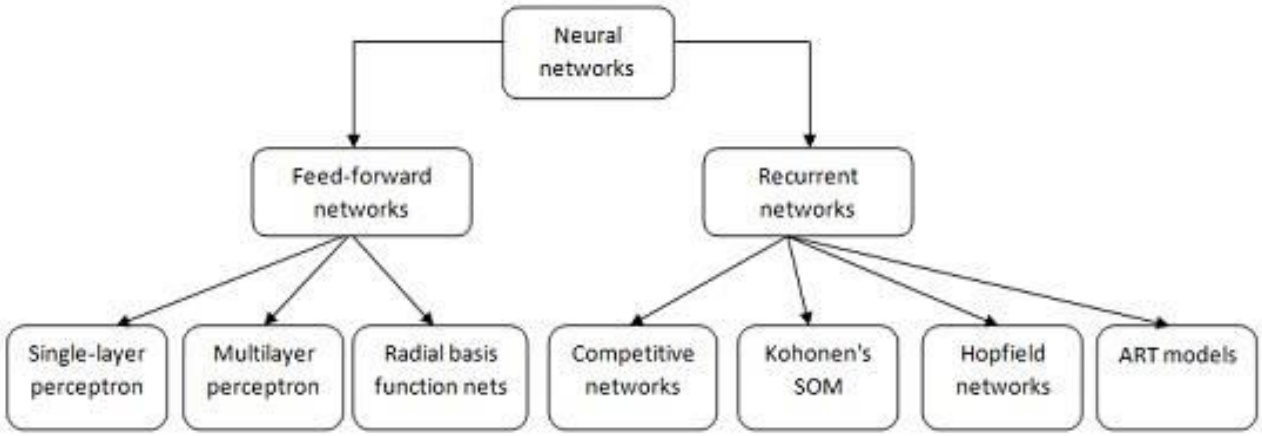


Figure 1.2 Neural Network Classification

Based on the connection pattern, the ANNs can be grouped in two categories:

1. Feed-forward network

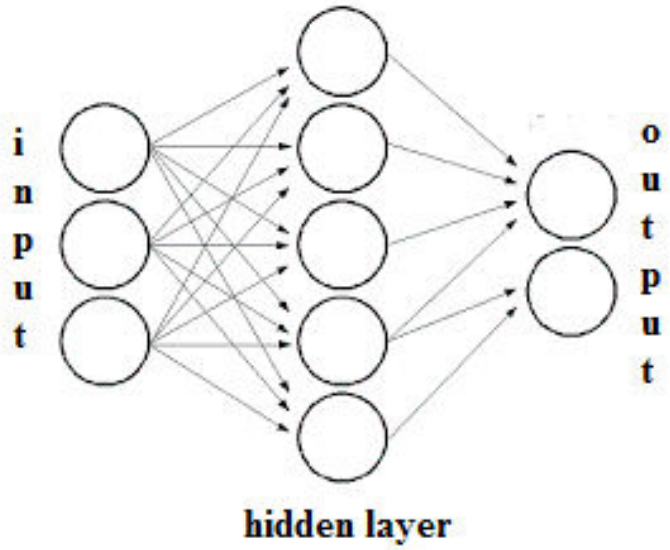


Figure 1.3 Feed-Forward Network

Figure 1.3 shows the structure of the feed-forward network. The feed-forward neural networks are most simple type of artificial neural network. In this network the information moves in only one direction that is forward direction. From the input neurons data goes through the hidden neurons and then finally to the output neurons. There are no cycles or loops in the network. Feed-forward networks are static, i.e. they produce or generate only one set of output dataset.

2. Recurrent network

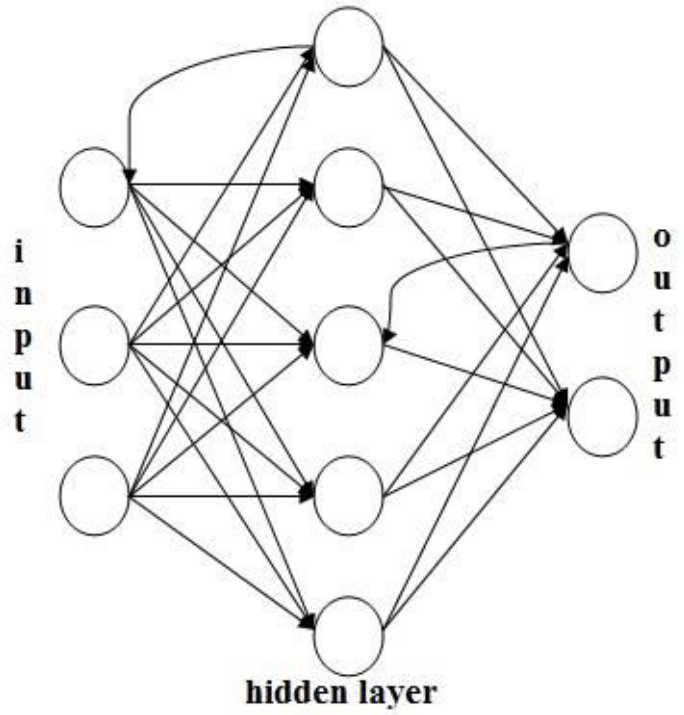


Figure 1.4 Recurrent Network

Figure 1.4 describes the recurrent network diagram. The prime feature of a recurrent neural network (RNN) is that the network consists at least one feed-back connection, so that the loops are created in the network. Recurrent networks are dynamic systems and are also called as feedback networks. When a new input pattern is given, output is obtained, but because of the feedback paths or loops, the input to each neuron is modified.

Deciding the number of hidden layers and its size (number of neurons) is one of the major task in designing the neural network. The capacity of the network increases with increasing the number of hidden layers or number of neurons in it. Capacity basically defines the amount of information a network can store. Neural network with more number of neurons becomes more complex. Smaller neural networks are preferred for less complex input datasets but smaller neural networks has their own disadvantages.

1.3 ACTIVATION FUNCTIONS

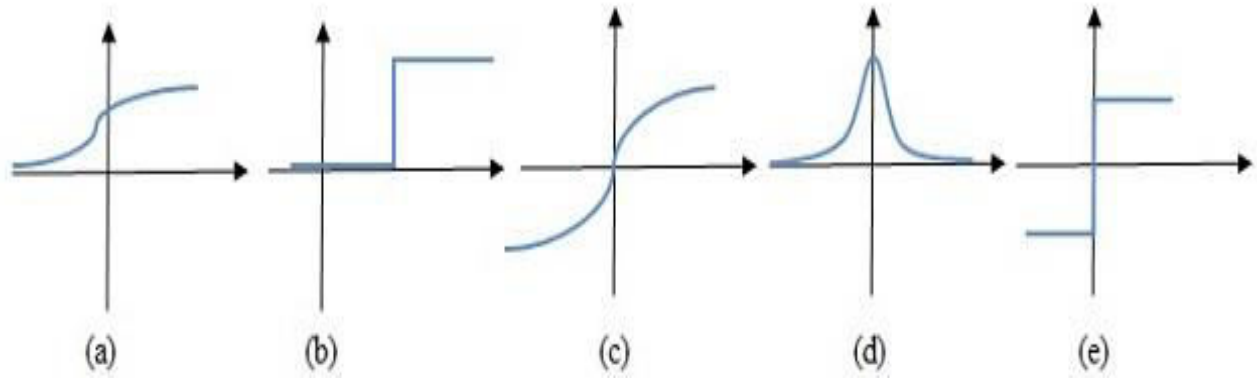


Figure 1.5 Different Types of Activation Functions: (a) sigmoid, (b) threshold, (c)hyperbolic tangent, (d) Gaussian, (e) signum.

ANN receives one or more than one inputs, sums them up and produces a output. To obtain this output, the sum of inputs is passed through some non-linear function which is called as activation function according to the need of the network. The applied activation function defines the level of the firing neuron. Some of the most commonly employed activation functions are sigmoid, threshold, signum, hyperbolic tangent, Gaussian etc.

Activation function makes neural networks able to deliver the required output as without activation functions ANN loses a huge power of its learning. It happens because the activation function applied are non-linear in nature, if they won't be applied, the system or network will remain linear regression which in turn will not modify or adjust the weights according to the need of the desired output. Therefore network will result in minimum extent of learning.

1.4 TYPES OF LEARNING

Different architecture needs appropriate learning algorithm to perform efficiently. training or learning is basically a mathematical procedure which adjusts and modifies the network weights according to the desired output and to perform the certain task. Learning ability is the prime trait of neural network and intelligence. Performance of the network improves by iteratively weight updating. The most important feature of ANNs is to learn from the examples. Which learning will be more appropriate is decided by the environment in which neural network is operating.

The main two learning algorithms are supervised and unsupervised. In case of supervised learning, the network is feed with a desired output for every set of inputs. Weights are accordingly updated to obtain the output as close as possible to the correct or desired output. An error function is generated by comparing the desired and obtained output which further is used to update the weights. The other type, that is unsupervised learning doesn't need the desired output feeding. It examines the underlying pattern in the data, and organize it into categories. It learns from the experience. Learning is further discussed in detail in chapter number 3.

1.5 INTELLIGENCE AND COMPLEXITY

There are number of learning algorithms for artificial neural networks, and these are algorithms are quite successful in executing simple tasks such as classification. The computers doesn't and can't think on its own like a human brain. The functioning and connections of human brain makes it really complex to implement it as any machine or computer. Being conscious about its surrounding is another trait which makes the human brain highly complex and any computer having this feature creates more complexity to itself.

What we can imply from this is that intelligence present in human brain is the result of complexity of size and structure. In order to achieve human brain intelligence in a computer, it is necessary that it meets the size and speed capabilities of human brain to some extent. The main challenge was in order to achieve a computer like that is the storage space. But the enormous growth and evolution of neural network makes it achievable. All that is needed now is the extremely fast learning algorithm. From the above explanation, the basic

requirements to achieve such a fast digital computer can be briefed down to two aspects, that are the storage space and speed. Artificial neural network can achieve it if an algorithm executing in parallel is applied.

1.6 BENEFITS AND DRAWBACKS OF MULTILAYER NEURAL NETWORK

The implementation of multilayer neural network provides the following beneficial attributes and capabilities:

- Learning- Artificial neural networks are capable of training and learning or we can they have ability to adapt without the help or assistance of the user.

- Input-output mapping- In supervised learning, every example is composed of a distinctive input signal and its corresponding desired output response. A sample is selected from the given training pattern and is fed to the network, in order to minimize the deviation of actual response obtained from the desired output, the coefficients of weights are rectified or changed. The neural network training is continued for different samples in the training set till the stable state of the network is attained. Hence networks learn through the training from the samples by establishing an input-output mapping for a particular case .

- Nonlinearity- Neurons are non-linear device. Resulting into a non-linear type of neural network.. Nonlinearity is extremely supreme attribute specially, when the input-output relationship is non-linear itself.

- Robustness- Multilayer feed-forward neural networks are very robust, which means their performance degrades smoothly as the level of increasing in the amounts of noise. Therefore they encourages to design a network which is immune to noise up to some extent.

There are some drawbacks of the neural network also exist:

- While employing sigmoid as activation function, the convergence of neural network becomes slow.

- The artificial neural networks cannot explain the processes that takes place during the training and learning of the network. This area of ANNs is still under research and development.

- The number of neuron weights in the neural nets is somewhat large due to large interconnections, which makes the network training quite slow.

1.7 MOTIVATION

1.7.1 Enormous Applications of Neural Networks

Neural networks are successfully being applied to many real world problems with obtaining efficient results. Some of the common areas are:

- Stock market prediction

- Fraud detection

- Price forecasts

- Medical diagnosis

- Treatment cost prediction

- Temperature prediction

- Industrial process and quality control

- Pattern recognition

- Signal processing

- Classification

and many more. There is further possibility of using neural nets in other real world areas. Exploring neural nets gives insight and ideas in that direction.

1.7.2 Exploration of a New Algorithm

A hybrid algorithm which employs the features of both supervised (LMS) and unsupervised learning (Hebbian) called as Hebbian-LMS algorithm is applied to the neural network. A form of LMS is created by the means of Hebbian which acts as unsupervised learning. It has application in practical engineering problems. It is one of the recently invented algorithm for neural networks which hasn't been explored in many areas. It is a unsupervised learning and expected to deliver success rate more than other algorithms in some cases. It is being expected that living neural networks may work on this algorithm.

Exploring this algorithm by applying to different feed-forward neural networks with different input datasets can give idea about this possibility of Hebbian-LMS algorithm being used by human brain for its functioning.

1.7.3 Growth of Machine Learning and Neural Networks

Machine learning and artificial intelligence have became an inevitable part of the technology now a days and the reason behind it is the vast application of these. All the major companies like Google, Microsoft, Facebook are using this technique and developing new applications.Neural networks are not something brand new, the concept was developed back in 1950s and many breakthrough happened in 1980 and 1990s. The 1950 and 1960s are called as the first golden age of the neural networks because of the successful invention of neuro-computer, which was initially developed for the pattern recognition only. There was no significance development was made in the 1970s but again in 1980s, the neural networks gained importance. In 1991, neural net with memory feature was developed.

2007 was the year when the neural nets were actually associated with the machine learning. In 2011, Microsoft introduced machine learning technologies into its speech recognition products, in 2012 Google used neural nets for recognition purpose and by 2013 used it in their speech recognition application along with the photo search. The successful development and modification of neural networks can be easily realized by the incident when Microsoft team outperformed a human being on the ImageNet competition in 2015.

1.8 ORGANIZATION OF THESIS

The thesis is organized as follows:

Chapter 1 presents a brief overview of neural networks and its features and machine learning.

Chapter 2 gives a brief description of the research that has been reported in literature in the field neural networks and learning paradigms, that is supervised and unsupervised learning.

Chapter 3 introduces basic concepts of learning algorithms and their modifications over the years.

C**hapter 4** presents structure and methodology used to obtain the results of the proposed algorithm.

Chapter 5 describes the experimental results and analysis of the proposed algorithm .

Chapter 6 discusses the conclusion of the report while also mentioning the future possibilities to carry forward the research in this domain.

CHAPTER 2

LITERATURE SURVEY

Artificial neural network and machine learning are interconnected fields. The ANNs are trained using different learning algorithms. From the past years there have been a noticeable growth in these two fields. This literature survey comprises the introduction, growth and applications of neural networks and machine learning algorithms.

2.1 INTRODUCING NEURAL NETWORKS

W.S. McCulloch *et al.*, 1943[1], proposed a theory which developed an analogy between the nervous system of living organism and the network that can be used for different calculus purposes. They described the network activity as "all-or-none" which is a characteristic behavior nervous system and the relations among them can be regarded as propositional logic. Various applications of the calculus are discussed. How neuron might works was explained by the means of neural network using simple electrical circuits.

D. O. Hebb, 1949[2], suggested several important ideas. The most famous is the one which has become the "Hebb" synapse. He assumed that the biological nervous system has "distributed" nature, which means a large number of cells participate in representation. He also postulated the formation of "cell assemblies" . The basic idea was that there were interconnections, self reinforcing subsets of neurons that formed the information system in the nervous system. Single cell might belong to more than one cell assembly. Multiple cell assembly could be active at once depending upon the nature of the task they are performing. This whole idea led to some very important inventions later, most common multi-layered neural network.

B. Widrow *et al.,* 1960[3], described the use of neural networks for adaptive filters and pattern recognition. The general pattern-recognition notion explained by the use of neural network proceeded by a trainable classifier. The neural networks can be trained to generate a set of values of outputs that are insensitive to rotation, translation, scale change, etc., of the retinal pattern. These outputs are disordered, but the adaptive layers whose weights can be altered are trained to make them in order and generate the actual patterns in “standard”

position. Multilayer adaptation algorithms are crucial in making such a scheme work, and they devised a new Madaline adaptation rule-MRII-for that purpose.

Daniel Svozil *et al.,* 1997[14], proposed back-propagation training algorithm for such networks. Also suggesting that the objective function's derivative which is obtained or calculated with respect to the threshold coefficients and weights are highly useful because in adaptation process of ANN, these derivatives play an important and valuable role. Further the applications of ANN in the field of chemistry were explored and analyzed along with their benefits and drawbacks, concluding that a proper analysis of the problem should be done before using neural network for it. There are other numerous options are available in place of neural networks in case of complex equation approximation. They also discussed the cases where the usage of neural nets for a particular prolems are inappropriate.

B. Widrow *et al.,* 2013[24], Encouraged by the life experiences, they proposed a completely new kind of computer memory. The central idea was taken from human memory. Neural network was employed to describe this memory called as cognitive memory. Cognitive memory does not work like a computer memory where specific data is stored in particular registers and retrieved by reading the contents of the particular memory register. Incoming data would be stored at the next available empty memory location, and indeed could be stored additionally at several empty locations. The stored data would neither have key words nor would it be located in known memory locations. Retrieval would be initiated by a prompt signal from a current set of inputs patterns. A search through the memory would be made to locate data that correlates to the prompt input. Neural networks are an important component of the human memory system, and their function is to retrieve information, not storage. The brain's neural networks are analog device, concerned with drift and sudden change. Only with persistent training, reliable action is possible. A cognitive memory is a learning system. Learning comprise storage of data in a cognitive memory. The unsupervised learning process is used cognitive memory.

2.2 LEARNING

B. Widrow, 1966[4], proposed bootstrap learning for linear threshold logic elements. Such elements are generally trained by applying input-pattern signals and the corresponding target binary responses. Often the target response to specific input patterns are not known. In such a

case bootstrap learning can be used to train the elements by deriving target from actual responses evaluated. Bootstrap adaptation is slower than conventional learning. Bootstrap learning is applied to the cases where conventional learning is not applicable. The basic objective of his research was to design an efficient algorithm for threshold-elements networks which are able to realize decision function and are not linearly separable.

Zhi-Quan Luo, **1991[7]**, considered the problem of training a linear feed-forward neural network by using LMS learning algorithm. The objective was to find a weight matrix for the neural network, by repeatedly exposing it to a finite set of examples, thus the sum of the squares of the errors is minimized. In this paper, he showed that, by decreasing the learning rate during each training cycle, the sequence of matrices produced by the algorithm converged to the optimal weight matrix. He also gave a general condition for the learning rates following which the LMS learning algorithm was guaranteed to converge to the ideal weight matrix.

M. Riedmiller *et al.,* 1993[8], presented a learning algorithm for multilayer ANN, termed as RPROP. In order to overcome the basic disadvantages of pure gradient-descent, RPROP was proposed. The weights were adapted locally according to the error function behavior. Unlike other learning algorithm where adaptation of weight is influenced by the size of the derivative, ROROP is only influenced by the temporal behavior sequence of signs of its partial derivatives, which results in efficient and transparent learning adaptation process. The learning step number reduced significantly as compared to the original gradient-descent adaptation procedure. The results obtained by using the new algorithm were promising in terms of robustness and convergence time.

Yuko Munakata *et al.,* 2004[15], described Hebbian learning as an algorithm having similarities with biological neurons and also as valid learning mechanism. Hebbian learning can be successfully implemented on to the neural level in terms of long-term potentiating (LTP) and long-term depression (LTD). Hebbian learning has self-organizing nature and is capable of extracting statistical regularities from the surrounding environment. So, Hebbian learning algorithms are biologically feasible, ecologically reasonable and powerful enough to consider for a number of behaviours across development. He presented that at the same time, a more stable error-driven learning mechanism is necessary. The most widely and commonly

used, back-propagation, is not ideal. More biologically feasible error-driven learning algorithms have been developed since back-propagation. These algorithms estimate error signals in a very similar way as in case of Hebbian learning and the ecological validity of these error-driven learning mechanism has been achieved to some degree. Therefore, Hebbian and error-driven algorithms may be combined into a unified framework. This unified framework can be more successful than either Hebbian or error-driving learning algorithms alone, in simulating a range of behaviors across perception, memory, language attention, and higher-level cognition. He also stated that further integration of learning algorithms, will likely to play a significant role in future progress in the modeling of development.

Zoubin Ghahramani, 2004[16], gave an overview of the unsupervised learning from the view point of statistical modelling. Unsupervised learning can be stimulated from information theoretic and Bayesian principles. He reviewed fundamental models in unsupervised learning, including factor analysis, state-space models, mixtures of Gaussians, and many variants and extensions. He derived the EM algorithm. The aim of his paper was to provide a high-level view of the unsupervised learning. The unsupervised learning can be understood within the framework of information theory and statistics. However, it has important role in ideas that influence neuroscience and psychology. Many of the models had been reviewed in this paper. These models were mainly inspired by the brain's ability to extract statistical patterns from sensory data and to recognize complex information. Unsupervised learning algorithms have ability to mimic some of the learning abilities of biological brains.

O. Brdiczka *et al.,* 2005[19], addressed the problem of supervised learning in intelligent environments in their paper. An intelligent environment perceives user activity and offers a number of services according to the recognize information from the user. An abstract context model was used to depict the intelligent environment. The context model includes the situations, roles played by network and relations between these networks. The aim was to adapt the system services, according to the changing needs of the user. For this, a supervisor gives feedback using an algorithmic learning method to rectify the system services that are found to be useless to user needs. The decision tree algorithm had been found to present the best results. Graph optimization was implemented in order to perceive the wide range of possible adaptation. The learning system verified the ambiguous choices by asking the

supervisor and the supervisor had the capability to correct the information when decisions of the learning system were not right.

B. Widrow *et al.,* 2012[23], introduced new learning algorithm for multilayer ANNs that was named as "No-Propagation" (No-Prop). In this algorithm, the hidden-layer neurons weights were set and fixed with random values. Only the output-layer neurons weights were trained, using LMS algorithm to minimize the mean square error. The No-Prop algorithm and the Back-Prop algorithm were compared and results showed that both the algorithms had the almost same performance when the number of training patterns are less than or equal to the network capacity. But when the number of training patterns is more than the capacity, Back-Prop had better performance. The No-Prop algorithm was much simpler and easier to implement than Back-Prop. Also, it converged much faster.

R. Sathya *et al.,* 2013[25], presented the comparison between supervised and unsupervised learning for pattern classification. They found that, the error back-propagation learning algorithm as provided by supervised learning model is coherent for numerous non-linear real-time problems, but KSOM of unsupervised learning model, offers better efficiency in classification.

B. Widrow *et al.,* 2015[28], presented the combined machine learning and named it as Hebbian-LMS algorithm. Hebbian learning is widely used in the fields of psychology and neurobiology and plays important role in neuroscience. The LMS (least mean square) algorithm is the most widely used adaptive algorithm, basic algorithm in the fields of signal processing, control systems, pattern recognition, and artificial neural networks. These are very different learning algorithm. However, a form of LMS can be constructed to perform unsupervised learning and, in this way LMS can be used in a natural way to implement Hebbian learning. Combining the two algorithm creates a new unsupervised learning algorithm that has practical engineering applications and provides insight into learning in living neural networks. They even proposed that the learning algorithm practiced by nature at the neuron and synapse level, may well be the Hebbian-LMS algorithm. Both, the Hebbian learning and LMS learning has important applications in their respective fields. At the same time, Hebbian-LMS is an unsupervised clustering algorithm that is very useful for automatic pattern classification.

CHAPTER 3

LEARNING

3.1 INTRODUCTION

The potential to learn is a crucial attribute of the intelligence. A learning process in case of neural network can be explained as the ability of network to update its architecture and weights of the connections so that it can perform specific task efficiently. Training patterns are available to make these connection weights learn in the neural network. Learning is an iterative process to improve the performance of the network by updating the connection weights. The iterative feature of machine learning is significant because when network model is introduced to some new data, they are able to adapt independently. The most fascinating trait of the artificial neural network is its ability to learn from the examples.

Machine learning has a application in many industries now a days. Industries working with huge size of data have identified the merits of machine learning technology. Few examples of industries utilizing the machine learning in best possible ways are:

- Financial services- Banks and other finance business industries are using the machine learning for one key purpose, that is to identify and prevent the fraud.

- Government- Government agencies like public safety and utilities has a high application of machine learning as they have several sources of data. Machine learning also helps in detecting fraud and minimizing the identity theft.

- Health care- By implementing the machine learning in the field of health, improved diagnosis have been achieved. Machine learning also contributes in number of wearable devices and sensors used for medical purposes.

- Marketing and sales- Suggesting the product based upon your previous purchase is achieved by using machine learning. Similarly promotion of products also imply use of machine learning.

- Oil and gas- From predicting refinery failure to finding new sources of energy involve implementation of machine learning.

To develop a learning process, firstly we should have a knowledge about the environment in which neural network is going to operate and all the information that is available to the neural network. The second important aspect is to understand that how weights of the neural network are going to be updated, which means knowledge about the learning rules which are going to be applied to the network, in simple words called as learning algorithm. There are some issues associated with the learning. The three cardinal and practical issues are: sample complexity, capacity and computational complexity. Capacity is associated to the amount of information a network can store. Sample complexity is related to the training pattern number required to train the neural network to assure a valid generalization. Computational complexity involves the requirement of time by a learning algorithm to obtain a output from training pattern. The main learning paradigms are supervised and unsupervised learning.

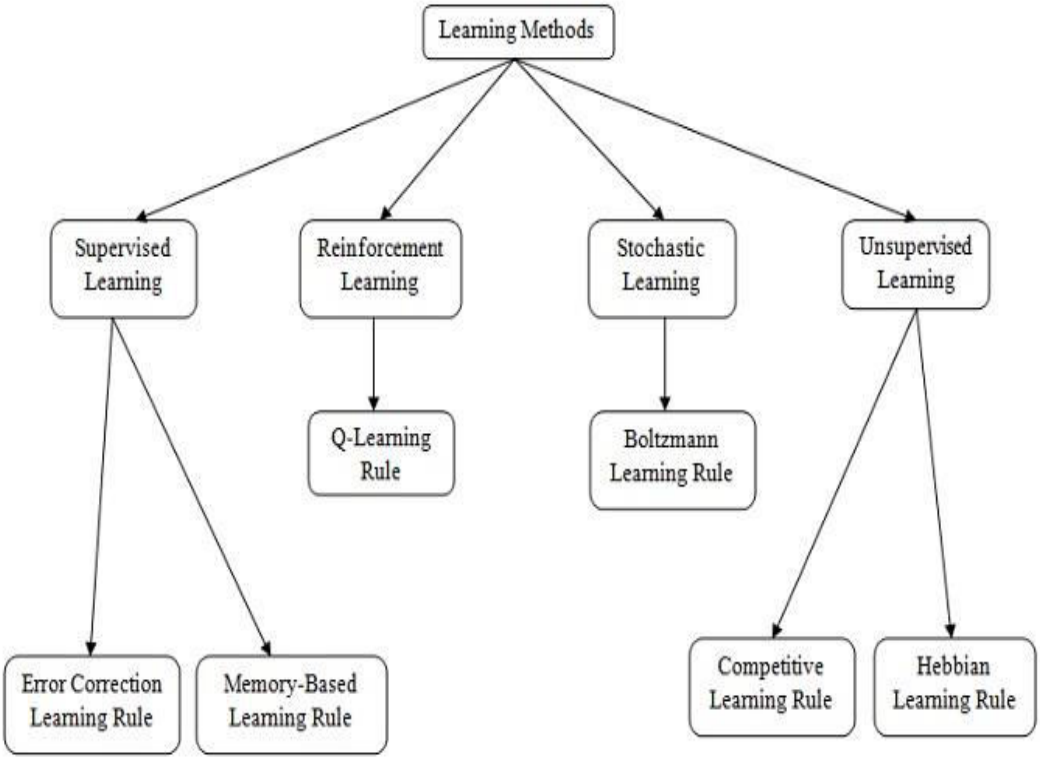


Figure 3.1 Learning Rules of ANN

ANN can be classified by different parameters such has: (a) based on type of interaction property (i.e. feed forward network or recurrent network); (b) based on its application (i.e. association, classification model etc.); (c) based on learning rule used (i.e. supervised, unsupervised or reinforcement learning rule). Mostly artificial neural networks are employed for pattern classification, as they posses efficient results for it because of its structure and learning methods. Figure 3.1 shows the different types of learning rules for ANN. The different learning rules are supervised, unsupervised and reinforcement learning rule. Supervised and unsupervised learning are discussed in detail in later sections whereas the reinforcement learning learns from error interactions and trials of its environment.

3.2 SUPERVISED LEARNING

In case of supervised learning, the classification of the data is already assigned. This learning rule can be successfully applied to the multilayer perceptrons or feed forward neural network. The network employing this learning rule must have three features:

One or more hidden layer, which is not the part of input or output layer. These hidden layers enables the network to learn and do the complex tasks.

- The network model must be fully connected.

- The non linearity exhibited in the neurons must be differentiable.

In supervised learning, algorithm produces an output in accordance with the desired output fed to the network during learning. This learning is quite common in pattern classification problems and commonly used in neural networks. In neural networks, the classification is used to calculate the error of the network and then adjust the network weights to reduce or minimize it. Learning through training in a supervised neural network is also called as error back-propagation algorithm. This error correction- algorithm trains the network using input-output samples and determines error signal. This error signal is the difference of the calculated output from the desired output fed to the network and then adjusts or modifies the synaptic weights of the neurons which is equal to the product of the error signal and the input of the neuron in neural network.

There are two types of data flow in supervised learning that is forward flow and backward flow:

- Forward flow- Input vector i.e. data is fed to the network, this input vector flows in the forward direction from layer to layer and neurons to neurons and produces the output at the end of the network. This calculated output is compared to the desired output and the error signal is generated for the neuron. The synaptic weights of the neurons remain the same during this propagation of the data.

- Backward flow- The generated error signal is propagated back to network. This backward propagation helps adjusting the neuron synaptic weights in order to make the calculated output as much as possible similar to desired response by reducing and minimizing the error iteratively.

Figure 3.2 shows the flowchart of the supervised learning rule in ANNs. First of all, input is determined that is what will be the input vector and the type of neural network is going to be implemented. Next step is to initialize the neural network parameters by some primary values, which can be random or defined values depending upon the need of the network task. After that the learning algorithm is performed iteratively in the forward flow and some synaptic weights are adjusted on the basis of error signal generated in the backward pass. This error signal is generated by comparing the obtained output and the desired response. This process continues until the algorithm completes its all iterations, the final output is stored. There are innumerable examples of supervised learning which can be implemented in neural networks and most commonly used is least mean square algorithm which was introduced by Widrow and Hoff in 1959.

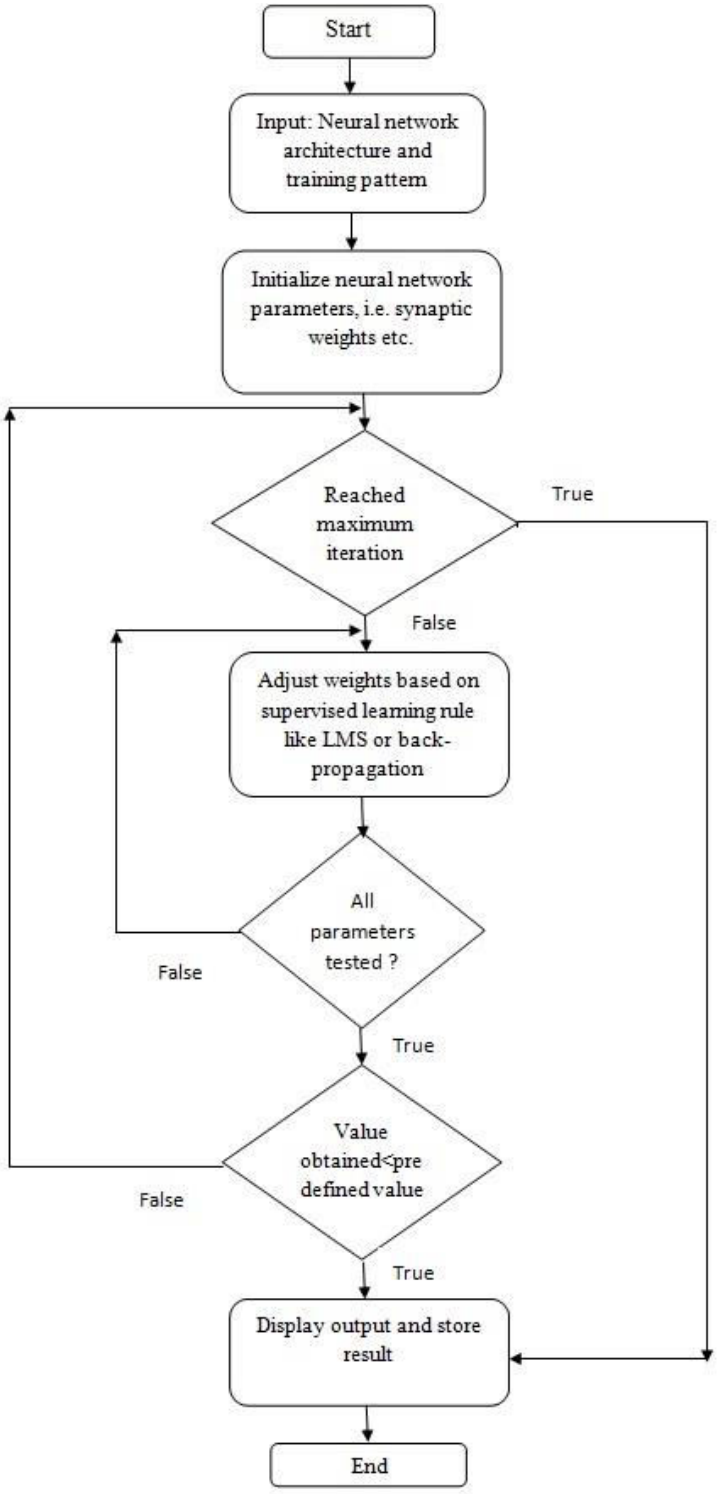


Figure 3.2 Flow Chart for the Supervised Learning Rule

3.2.1 LMS and its Variants

LMS is an adaptive algorithm, works on the principle of method of steepest decent. LMS is comprised of iterative procedure which makes adjustment of the synaptic

weights in order to minimize the mean square error. It is one of the simplest algorithm to implement. The weight vector is initially given some random and arbitrary values which is modified and adjusted iteratively to reduce or minimize the error. LMS can be explained by the three basic equations:

$$\text{OUTPUT: } y_k = X_k^T W_k \tag{3.1}$$

$$\text{ERROR: } e_k = d_k - y_k \tag{3.2}$$

$$\text{WEIGHT: } W_{k+1} = W_k + 2\mu e_k X_k \tag{3.3}$$

Equations from 3.1 to 3.3 defines the LMS algorithm mathematically. X_k is the input vector, W_k is the weight vector, k defines the number of iteration, e_k is the error vector, d_k is the desired response or target output and W_{k+1} is the weight vector which is to be used as W_k in the next iteration, that is it the modified or adjusted weight in accordance with the error signal generated in the previous iteration. Here μ is the convergence factor which decides the convergence speed of the LMS algorithm. LMS has mainly two variants:

1. Normalized LMS

In normalized LMS the weight update equation that is equation 3.3 is modified as:

$$W_{k+1} = W_k + 2e_k W_k \left[\frac{\mu}{\|X_k\|^2}\right] \tag{3.4}$$

In equation 3.4, $\|X_k\|$ is absolute value of the input vector and usually denoted as $\text{norm}\|X_k\|$. Instantaneous samples can suppose any value of $\text{norm}\|X_k\|$ which should be very large. So the only difference between LMS and NLMS the change in convergence factor of NLMS which becomes $\left[\frac{\mu}{\|X_k\|^2}\right]$ instead of μ as in case of LMS. In this way the, for the next iteration, the weight vector changes by minor value. The effect of huge fluctuations in the power levels of the input signal is rectified at the adaptation level. It works on the principle that when the new input dataset is applied the parameters of the system should be disturbed or changed to minimal fashion. So it

can be concluded that the NLMS offers better results as compared to LMS but it is more complex than LMS.

2. Sign LMS

Time is a crucial constraint of any system, sign LMS provides fast adaption to the system. Mathematically it can be seen as:

$$\text{sgn(a)} = \begin{cases} 1, & ; a > 0 \\ 0, & ; a = 0 \\ -1, & ; a < 0 \end{cases} \tag{3.5}$$

$$W_{k+1} = W_k + \mu\, X_k\, \text{sgn}(e_k) \tag{3.6}$$

Equation (3.5) and equation (3.6) defines the modification of LMS to obtain the sign LMS. Sign algorithm is implemented to minimize the error. The advantage of sign algorithm is it makes the adaptation fast by reducing the error significantly for the next iteration and the drawback is that the convergence rate is decreased.

3.3 UNSUPERVISED LEARNING

Unsupervised learning is much harder to implement as compared to supervised learning. It works on the principle to make the computer or network learn how to do something without providing the target. It is said to be self organizing process and seems plausible in the field of biology and neurology. In this, the network decides itself that what would be the best outcome for a particular set of input signal because we provide no target or the external source for comparison. The unsupervised learning introduces the ability of a network to learn and organize information without provision of any error signal to determine the potential solution. in unsupervised learning, it is possible to develop a framework based on the purpose that the machine's goal is to build and create representations of the particular input that can be employed for decision making, predicting future inputs etc. This learning algorithm makes the network learn and then use those learned aspects on the other incoming input datasets. The most widely used unsupervised learning is Hebbian learning rule.

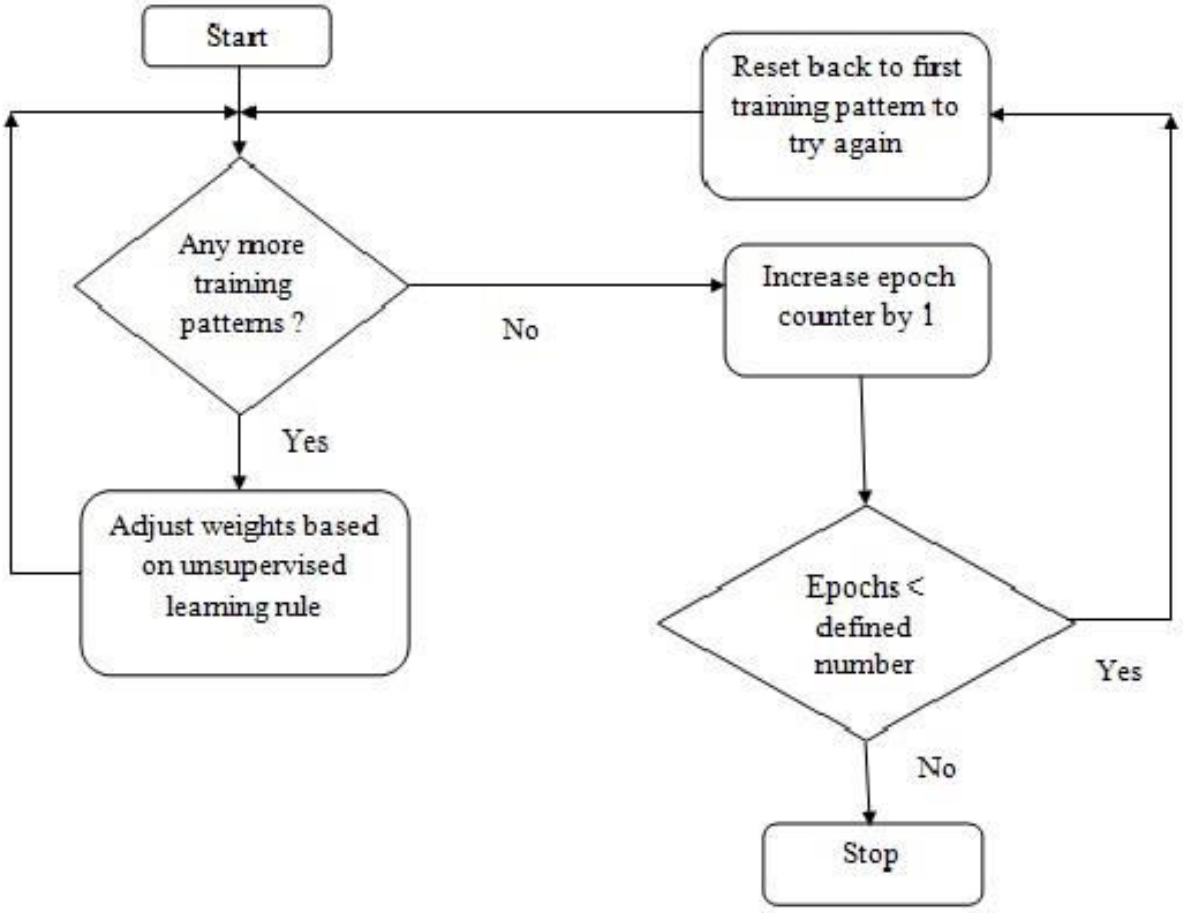


Figure 3.3 Unsupervised Learning Rule Flow Chart

Figure 3.3 shows the flowchart of unsupervised learning. First of all the network is presented with training patterns which can be called as input vector, Some weight adjustment unsupervised learning rule like Hebbian or competitive algorithm will be applied in order to adjust the weights. The process of presenting training patterns continues until all the patterns are not fed. Number of training pattern decides the number of epochs the algorithm must run by. When the network run by the complete number of defined epochs, the algorithm stops. The network is supposed to learn from the presented pattern when all the epochs are completed. When presented with test data, network must be able to classify it in the same manner as trained by the earlier presented training pattern.

3.3.1 Hebbian Learning Rule

The neurophysiologist Donald Hebb proclaimed that the repeated excitation of biological neurons facilitates the communication between them. Then this idea was further expanded to the artificial system, which gave the idea of artificial neural system. When two neurons of the ANN are fully connected through the weights, Hebb's principle will increase the common weight when there will be flow of information between them. If the initial weight is positive, the Hebbian learning makes it more positive in further training whereas if weight is initially negative, it becomes more negative only. It works on the principle of 'units that wire together fire together'. In short, if the firing of one neuron frequently leads to the firing of another, the influence of the first neuron increases on the second, thus increasing the possibility that the first neuron will cause the second neuron to fire. It is the oldest learning rule and also known as Hebb's postulate of learning. The change in synapse weight only depends on the activities of the two neurons connected by it. This simplifies the complexity in a VLSI implementation while performing learning. It is an unstable learning rule because produces unbounded weights but when normalized, it can become highly useful. Trained neural network produces an output which saves the maximum information from the presented input as required for signal representation. If the sign of the Hebbian update is reversed, a network can be obtained which produces the input from the outputs.

Let us suppose that there are two neurons, namely x_i and x_j with corresponding weight w_{ij} then Hebbian rule can be written mathematically as:

$$w_{ij}(k+1) = w_{ij}(k) + \eta x_i(k) x_j(k) \tag{3.7}$$

Here k defines the number of iteration and η is the learning parameter. This equation simply describes that when both the neurons x_i and x_j will be firing, i.e. have value 1 at any iteration k then the value of $w_{ij}(k+1)$ will be changed. And if one or none of the neuron will be firing that is having value 0, the addend on the right side will remain 0 and no weight adjustment will take place. Because Hebb's rule only causes the increasing of weights in a network, it has the problem of increasing weights

without any bound, and thus it is unstable. The solution to this problem is to set a bound for increase in weight and Oja's rule is a common method of weight normalization, and this modification makes the Hebb's rule stable:

$$w_{ij}(k+1) = w_{ij}(k) + \eta x_j(k)[x_i(k) - w_{ij}(k)x_j(k)] \quad (3.8)$$

It can be seen that Oja's rule is applicable learning in one direction, when weights are either adjusted up or down, but not in both directions. There is another way of making sure that weights do not saturate to their upper bounds, or approach infinity and that is to enable some amount of decay to the weights over iterations. Mathematically this decay component can be given by:

$$\gamma\big(1 - x_i(k)x_j(k)\big) \quad (3.9)$$

Therefore, if either x_i or x_j are non-firing, then decay is equal to γ . This can be append in the original Hebbian rule as:

$$w_{ij}(k+1) = w_{ij}(k) + \eta x_j(k)\, x_i(k) - \gamma(1 - x_i(k)x_j(k)) \quad (3.10)$$

Equation (3.10) shows the final normalization of the weight in Hebbian rule which bounds the weight increasing in upward directions and hence makes the Hebbian rule stable.

3.3.1.1 Anti Hebbian learning rule

Anti-Hebbian is the inverse of Hebbian rule, that is the weight decreases without bound. Mathematically, it is opposite of the Hebbian rule and can be written in a equation as:

$$w_{ij}(k+1) = w_{ij}(k) + \eta x_i(k)\, x_j(k) \quad (3.11)$$

The behavior of the anti-Hebbian is defined as decorrelating.

3.3.1.2 Competitive Hebbian learning

Competitive Hebbian learning overcomes the weighting problems of standard Hebbian learning by treating the neuron connections as competitors for impact. Problem of unregulated weight adjustment is overcome in competitive Hebbian learning by adding weights that causes the increased efficiency of the neural network, and by fixing the weights before training. In other words, competitive Hebbian learning doesn't adjust or modify the weights in a network. Rather, a random weight vector is generated before training corresponding to some probability distribution, and these weights are added or removed in order to minimize or reduce the error at the single neuron level.

3.3.2 Hebbian-LMS Learning Algorithm

Recently discovered unsupervised learning and a combination of two learning paradigms, this learning algorithm named after both the learning algorithms as Hebbian-LMS. It is an implementation of Hebbian learning by the means of the LMS algorithm and it has many practical engineering applications with an assumption that this is the nature's algorithm performed by the biological neurons to interact with each other. Mathematically too, it is a combination of Hebbian and LMS and can be represented as follows:

$$\text{WEIGHT: } W_{k+1} = W_k + 2\mu e_k X_k \tag{3.12}$$

$$\text{ERROR: } e_k = \text{SGM}\left(X_k^T W_k\right) - \gamma X_k^T W_k \tag{3.13}$$

Equation (3.12) and equation (3.13) explains the error vector and weight updated vector generation. It can be seen from these equations too that it is a combination of Hebbian and LMS as weight adaptation is done by using Hebbian and using LMS an error signal is also being generated. The question arises is that even after the error signal generation why is an unsupervised learning? Well the answer to this question is that the output of the represented input data to algorithm is not known. This represented input data works as training pattern only which generates an output which help network learn from. The error signal does not contain any term which gives the

target output. Therefore the error is not a function of desired response therefore it's not a supervised learning. The final output can be given by:

$$(\mathrm{OUTPUT})_k = \begin{cases} \tanh(X_k^T W_k), & X_k^T W_k \geq 0 \\ 0, & X_k^T W_k < 0 \end{cases} \tag{3.14}$$

Hebbian-LMS works with all configurations of networks. The network can be layered structure or it can be interconnected in random configurations.

3.4 SOME OTHER BASIC TYPES OF LEARNING RULES

There are some other types of learning rules: error correction, Boltzmann and competitive learning, which are being used in different applications as required by the specific task.

3.4.1 Error Correction Rule

In the supervised learning, the network is fed by a desired response for each input pattern. During the process, an actual output y is generated which may or may not be equal to the desired output d. When using error correction rule, the error signal e equals to (d - y) would be generated. This generated error signal is used to modify the weights which gradually reduces the error value in the network and hence gives the final output very close to the desired output. Learning only occurs when there will be error signal generation. The "back propagation learning algorithm" is also based upon the error correction rule.

3.4.2 Boltzmann Learning Rule

Boltzmann machines are recurrent networks which are symmetric and they consists binary units that is they represent +1 for the "ON" condition and -1 for the "OFF" condition. Let's assume there are two neurons names neuron i and neuron j, by symmetric it means that the connection weight from neuron i to neuron j will be same as connection weight between the neuron j to neuron i. A subset of neurons which interact with the environment is called as visible while the other which do not interact is called as hidden. Each neuron generates an output according to the Boltzmann

distribution. Boltzmann learning is stochastic learning rule. The connection weights are adjusted in such a way that the state of visible units satisfies to some desired probability distribution. Boltzmann learning can be seen as the special case of the error correction learning rule where error is not directly calculated by subtracting the actual output from the desired output, but calculated as the difference between the correlations among the two output neurons.

3.4.3 Competitive Learning Rule

Competitive learning output neurons compete with each other for activation which results in only one active neuron at a time. This learning rule is found to exist in the biological neural system. Data clustering or categorizing of input data takes place. Similar data is represented under the single unit. This grouping is performed automatically on data correlation basis. The active unit at a particular time can be termed as winner and only it's weight get updated. The objective of this learning rule is to move the stored pattern in the winner unit closer to the input pattern. The learning process continues until the learning rate becomes zero. This causes the instability in the system or network. So in order to achieve stability, the learning rate is forced to decrease gradually during the learning process.

CHAPTER 4

STRUCTURE AND METHODOLOGY OF PROPOSED ALGORITHM

4.1 BASIC NETWORK STRUCTURE

This chapter describes the method of constructing and training the neural network used by proposed algorithm. The network described is feed forward network, consisting of three layers named input, hidden and output layer. It can have more than one hidden layer. Each layer having different number of neurons and each neuron has an activation function. Some of the commonly used activation functions are identity function, binary step function, sigmoid function, binary sigmoid, bipolar sigmoid etc. This activation function remains the same for each neuron of a particular layer which specifies the output of the neuron for the given input. In this case we have used the hyperbolic tangent and sigmoid activation functions.

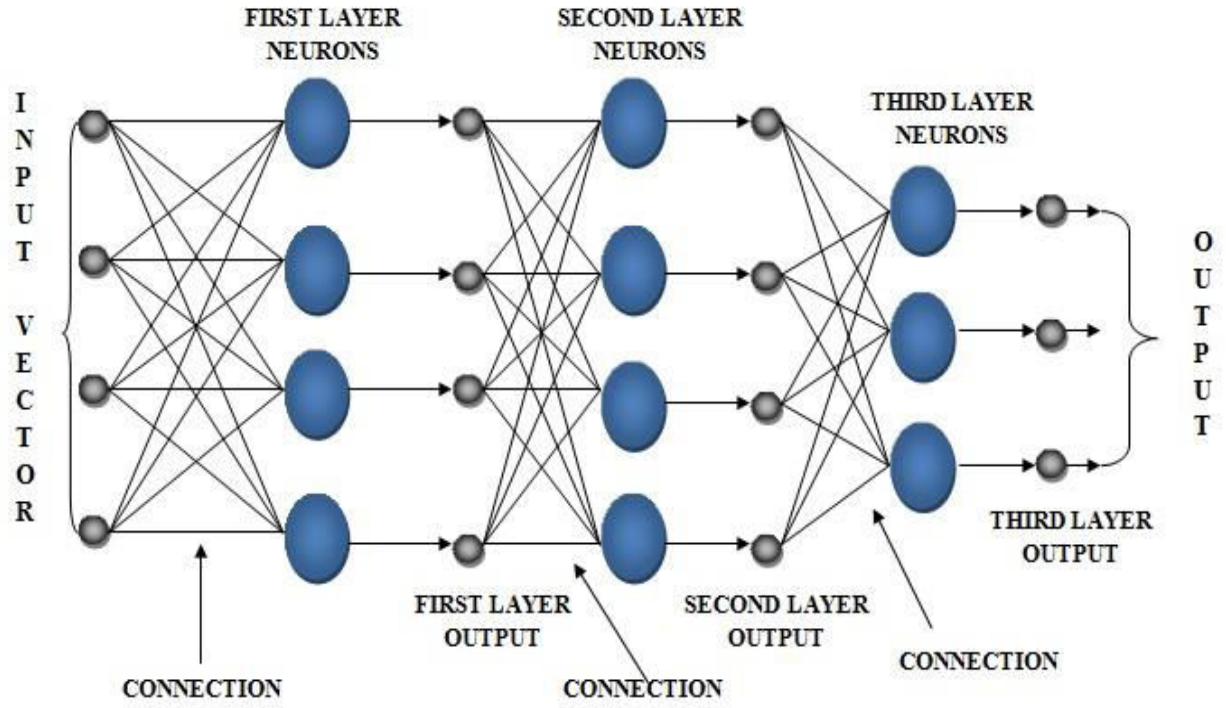


Figure 4.1 Multilayer Neural Network

Figure 4.1 shows a multilayer neural network with three layers consisting different number of neurons. The network shown is 4-4-3 network i.e. the input layer(first layer) and hidden layer(second layer) has 4 neurons and output layer(third layer) has 3 neurons. Such network is trained using the proposed algorithm. The number of neurons in the layers were modified or changed according to the dataset used for training. The operation performed on these synapses are parallel. Parallel operation simply means that the weights are adjusted or adapted concurrently, which increases the speed of convergence of the complete network. If these layers were trained until their individual convergences, i.e. the second layer synapse adaptation starts after the convergence of the first layer and third layer adaptation commences after the convergence of second layer then the time taken for training would be three times of the single neuron training. Parallel operation trains all the layers simultaneously and makes the network faster.

The proposed algorithm uses the combination of two learning rules, i.e. Hebbian-LMS and LMS(least mean square). The hidden layer is trained using Hebbian-LMS learning rule which is an unsupervised learning whereas the output layer is trained by LMS learning rule which is a supervised learning. The developed algorithm acts as a supervised learning.

4.2 HEBBIAN-LMS LEARNING RULE

Figure 4.2 is a diagram of a neuron whose weights are trained with Hebbian-LMS learning. X_k is a input vector and W_k is the weight vector. The error function e_k is characterized by the following signal in the figure above,

$$\text{ERROR: } e_k = \tanh\left(X_k^T W_k\right) - \gamma X_k^T W_k \tag{4.1}$$

The weight vector is modified by the equation:

$$W_{k+1} = W_k + 2\mu e_k X_k \tag{4.2}$$

In this algorithm, there is no desired response is fed for each input pattern, therefore it acts as unsupervised learning. The parameter μ controls the speed of convergence and stability. The

parameter μ performs the same function as in case of LMS algorithm. γ can have any positive value, but it has to be less than the initial slope of the activation function used.

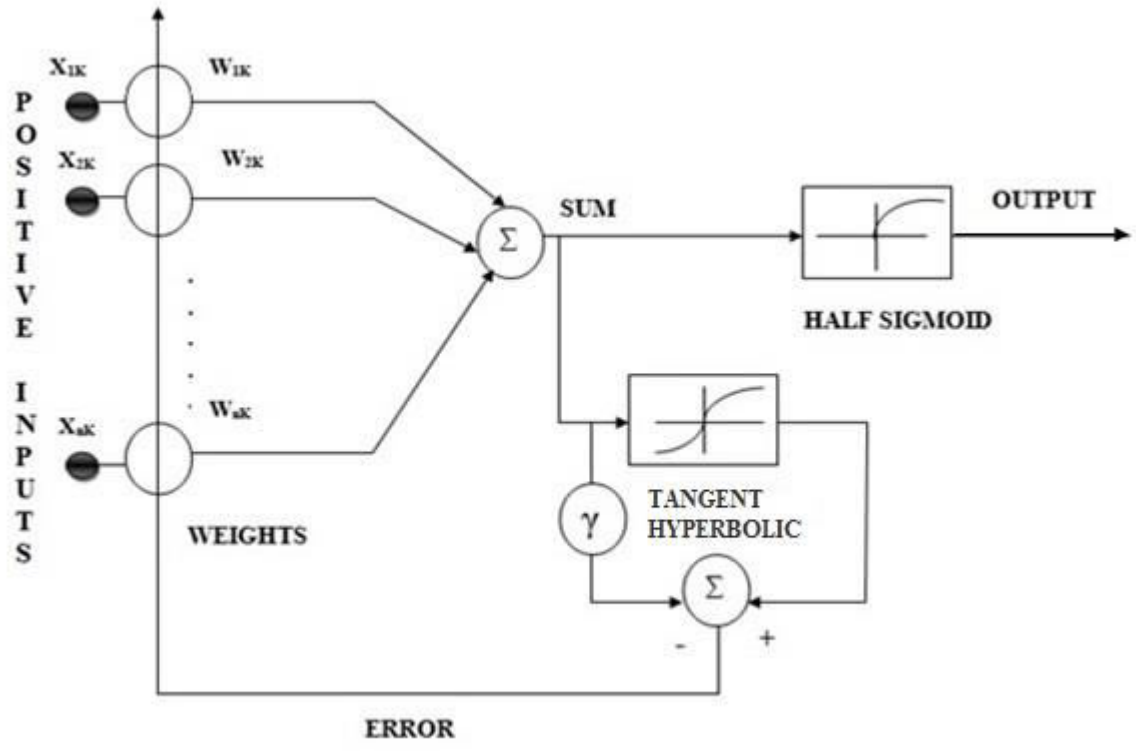


Figure 4.2 A Neuron Trained With Hebbian-LMS Learning

The final output of the neuron is obtained by passing the signal from the half sigmoid function.

$$(\mathrm{OUTPUT})_k = \begin{cases} \tanh(X_k^T W_k), & X_k^T W_k \geq 0 \\ 0, & X_k^T W_k < 0 \end{cases} \tag{4.3}$$

Equation (4.1) and equation (4.2) represents the training procedure for the weights (synapses) and equation (4.3) describes the final output through the neuron, which limits the hidden layer output between 0 to +1. All the synaptic weights are initially set to random values, and then adjustment of the weights begins by applying the Hebbian-LMS algorithm independently to all of the neurons and their input synapses.

4.3 LMS LEARNING RULE

Fig 4.3 shows the diagram of a neuron implementing LMS learning algorithm. The notations used are similar to the previous one, i.e. X_k is input vector to output layer which now the output of the hidden layer,, W_k is weight vector and e_k is the error. The error is the difference between the desired response (d_k) and the output generated. Generated output is the product of the input X_k and its corresponding weight W_k.

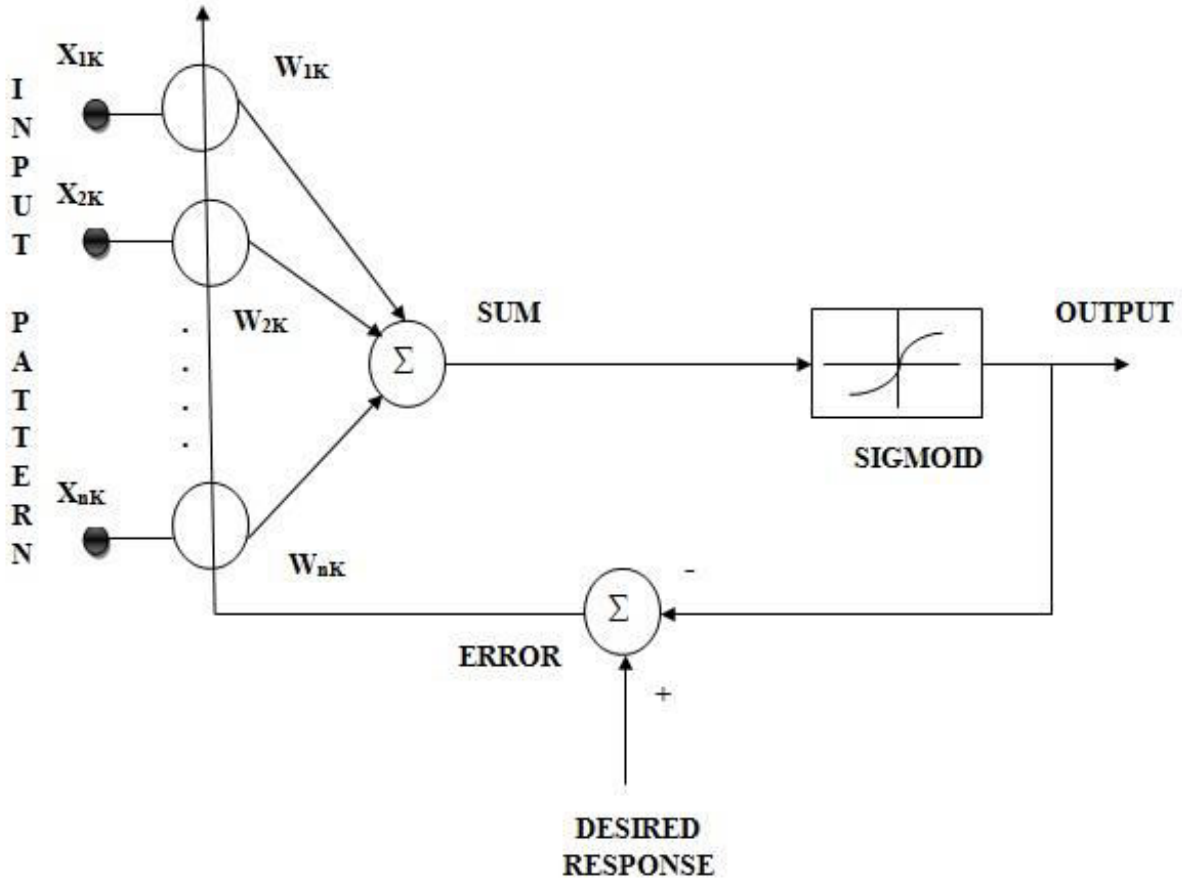


Figure 4.3 A Neuron Trained With LMS Algorithm

$$\text{OUTPUT} = y_k = X_k^T W_k \tag{4.4}$$

$$e_k = d_k - \tanh(X_k^T W_k) \tag{4.5}$$

$$W_{k+1} = W_k + 2\mu e_k X_k \quad (4.6)$$

The weight adaptation commences by the equation (4.6). The equation (4.4), equation (4.5) and equation (4.6), all together comprises the LMS learning. LMS learning is implemented to train the output layer of the used neural network in the proposed algorithm.

The output layer received its input from the hidden layer neurons after performing the Hebbian-LMS training algorithm on it. The half sigmoid function is applied to introduce the non-linearity and to obtain the output from the hidden layer which is to be given to the output layer as input. The half sigmoid function simply limits these values between 0 and 1. The LMS converges the weights to the optimum weights. These weights are updated by calculating the gradient of the mean square error by equation (4.5). The value of mean square error gradient assist whether the value of weights should be increased or decreased further. If the value of error keep increasing positively, it implies that the same weight which is being used will further increase the error. In order to reduce this error, weights need to be reduced. Similarly, if the error is negative, the weights need to be increased reduced. The weights are revised by using the equation (4.6).

CHAPTER 5

EXPERIMENTAL RESULTS AND ANALYSIS

5.1 PROPOSED ALGORITHM RESULTS

The proposed algorithm is a new kind of learning algorithm, in which we implemented Hebbian-LMS and LMS learning rules together to train hidden layer and output layer respectively in the neural network. Hebbian-LMS is unsupervised learning in nature whereas the LMS is supervised, and the proposed algorithm behaves as supervised learning. MATLAB tool is used for different analysis.

The analysis of the proposed algorithm is done for different cases considering different conditions. Following are the classifications on which basis the analysis is performed :

3. For different standard datasets, i.e. Iris, Heart disease, Gesture phase and Breast cancer datasets. These datasets are obtained from UCI repository.

4. For different number of hidden layer neurons, initially starting from number of attributes and gradually increasing to 10, 20, 50 and 100.

5. For different number of epochs, that is 100, 1000 and 2000.

6. Mean square error(MSE) comparison with back-propagation learning algorithm.

Table 5.1 shows the average success rate of the proposed algorithm when iris and breast cancer datasets are given as input data. The analysis is performed for different number of hidden layer neurons with varying epochs. It can be observed that success result increases with increase in hidden layer neuron numbers and epochs. The lowest results are obtained with lowest number of hidden layer neurons and epochs which is equal to number of attributes and 100 respectively.

Table 5.1 Average Percentage Success Rate of Iris Dataset and Breast Cancer Dataset

DATASETS	IRIS					BREAST CANCER				
HIDDEN LAYER NEURONS	4	10	20	50	100	9	10	20	50	100
100 EPOCHS	58.1	75.9	84.5	89.2	94.7	70.872	77.595	78.121	78.869	78.829
1000 EPOCHS	61.9	76	85.1	90.1	95.1	76.136	77.910	78.497	78.926	78.955
2000 EPOCHS	68.6	78.2	87.6	92.7	95.6	77.781	78.200	78.869	78.983	78.969

For iris dataset, hidden layer neurons 4 and epochs 100, the result obtained is 58.1% and for the same neuron number and 2000 epochs, success rate increases to 68.6% . Whereas for breast cancer dataset minimum hidden layer neurons are 9 (which is the number of attributes of breast cancer dataset) and epochs 100, and the result calculated is 70.872%. On the other hand with same number of neurons but increased epochs, i.e. 2000 the percentage success result increases to 77.781%. The same scenario can be seen for other number of neurons too in both iris and breast cancer dataset. The best results are obtained for maximum number of neurons and epochs used, i.e. 100 and 2000 respectively, which is 95.6% for iris and 78.696% for breast cancer dataset.

Table 5.2 Average Success Rate of Heart Disease Dataset and Gesture Phase Dataset

DATASETS	HEART DISEASE				GESTURE PHASE			
HIDDEN LAYER NEURONS	13	20	50	100	18	20	50	100
100 EPOCHS	80.940	82.129	82.820	84.234	82.852	83.789	90.374	94.294
1000 EPOCHS	81.529	82.234	83.234	84.882	83.055	84.066	90.741	94.300
2000 EPOCHS	82.764	83.023	85.234	84.940	83.748	84.677	91.135	94.417

Table 5.2 shows the same analysis as in table 5.1 only with the different datasets, i.e. heart disease and gesture phase datasets. For heart disease dataset the percentage success rate increases from 80.940%(when hidden neuron=13, epochs=100) to 84.940% (when hidden neuron=18, epochs=2000). Same can be observed with gesture phase dataset. The similar pattern of increasing percentage success rate with increased number of hidden layer neurons and number of epochs can be observed.

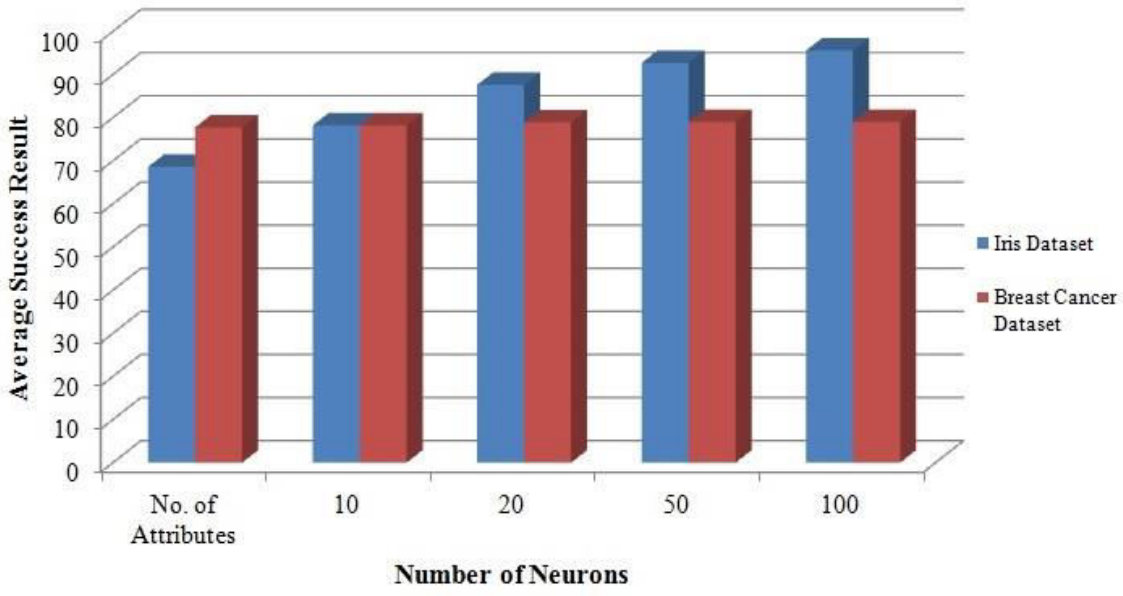


Figure 5.1 Bar Graph of Success Rate vs. Number of Neurons of Iris and Breast Cancer Datasets

Figure 5.1 and figure 5.2 shows the bar graphs for datasets for success rate vs. number of neurons. The results are plotted in the bar graphs are for 2000 epochs only. It can be observed in both the plots that as the number of hidden layer neurons increases from attribute numbers to 100, the rate of success increases gradually. Iris, breast cancer, heart disease and gesture phase datasets has 4, 9, 13, 18 number of attributes respectively. From the bar graphs, it can be inferred that the training is more successful when the hidden layer has more number of neurons and the algorithm is run for more number of epochs.

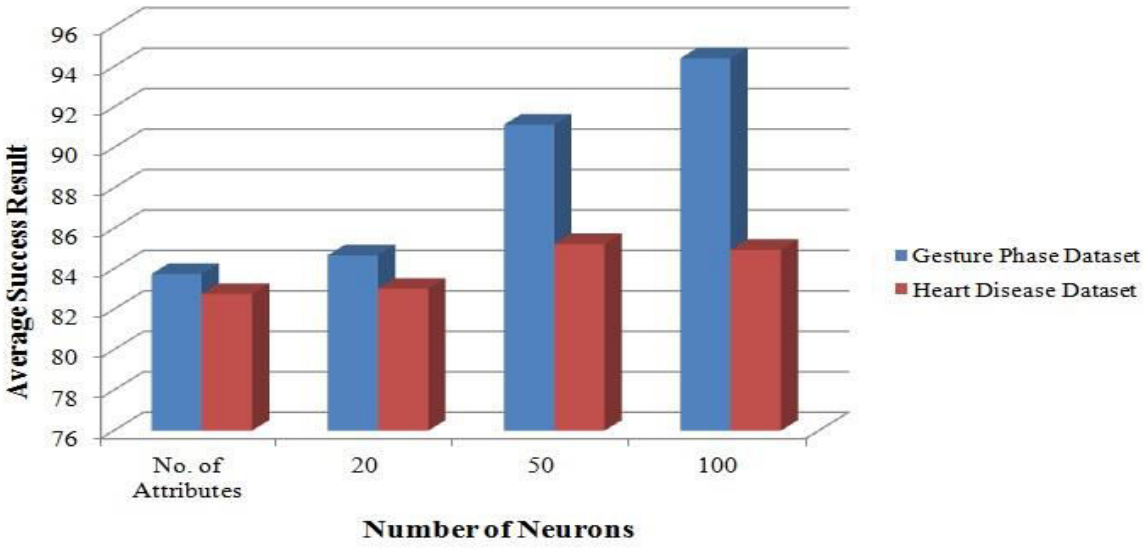


Figure 5.2 Bar Graph of Success Rate vs. Number of Neurons of Gesture Phase and Heart Disease Datasets

Table 5.3 and table 5.4 shows the comparison of the average of the mean square error (MSE) calculated using proposed algorithm and back propagation. The number of epochs are the same for all hidden layer neurons that is 2000 epochs, starting from the hidden layer neurons equal to the number of attributes gradually increasing to 10,20,50 and 100. It can be seen that the average MSE obtained for proposed algorithm is less than the MSE obtained by back propagation.

Table 5.3 Comparison of MSE (average) Obtained

DATASETS	IRIS					BREAST CANCER				
HIDDEN LAYER NEURON	4	10	20	50	100	9	10	20	50	100
PROPOSED ALGORITHM	0.005	0.005	0.029	0.013	0.015	0.019	0.019	0.018	0.017	0.018
BACK PROPAGATION	0.035	0.052	0.103	0.040	0.041	0.032	0.026	0.034	0.032	0.032

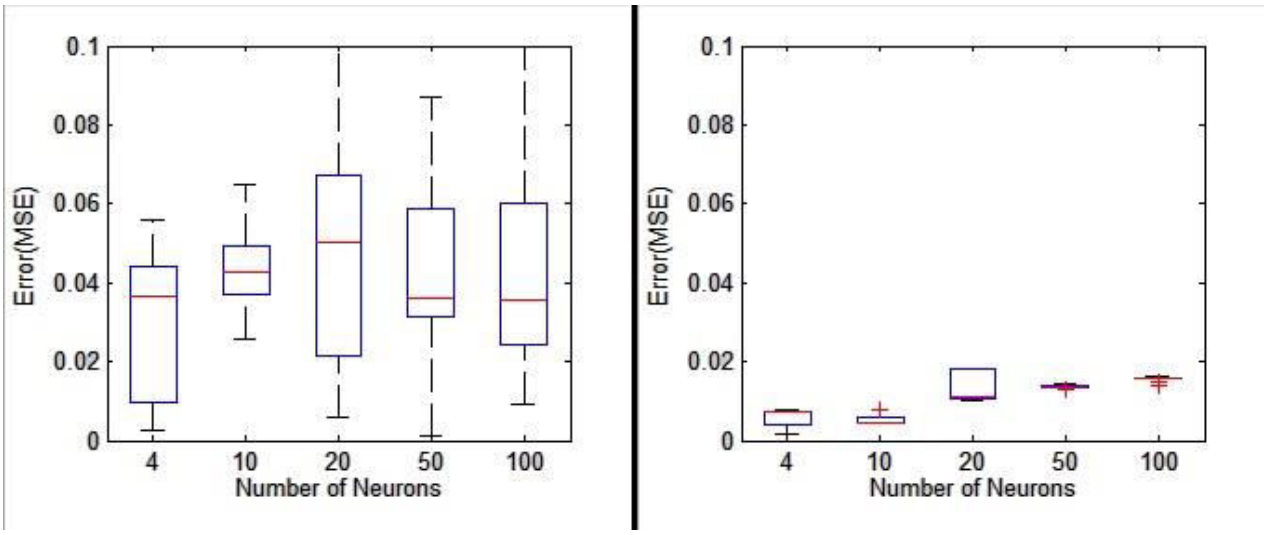
Table 5.4 Comparison of MSE (average) Obtained

DATASET	GESTURE PHASE				HEART DISEASE			
HIDDEN LAYER NEURON	18	20	50	100	13	20	50	100
PROPOSED ALGORITHM	0.0048	0.0041	0.0072	0.0075	0.0155	0.0167	0.0244	0.0478
BACK PROPAPGATION	0.0348	0.0391	0.0307	0.0230	0.1039	0.1115	0.1171	0.0133

From figure 5.3 to figure 5.6 shows the box plot of the mean square error calculated. The (i) in all figures shows the MSE calculated by using back propagation algorithm and the (ii) of all figures shows the MSE computed by the proposed algorithm. The MSE is computed for different number of neurons in the hidden layers starting from the number of attributes of the datasets for 2000 epochs.

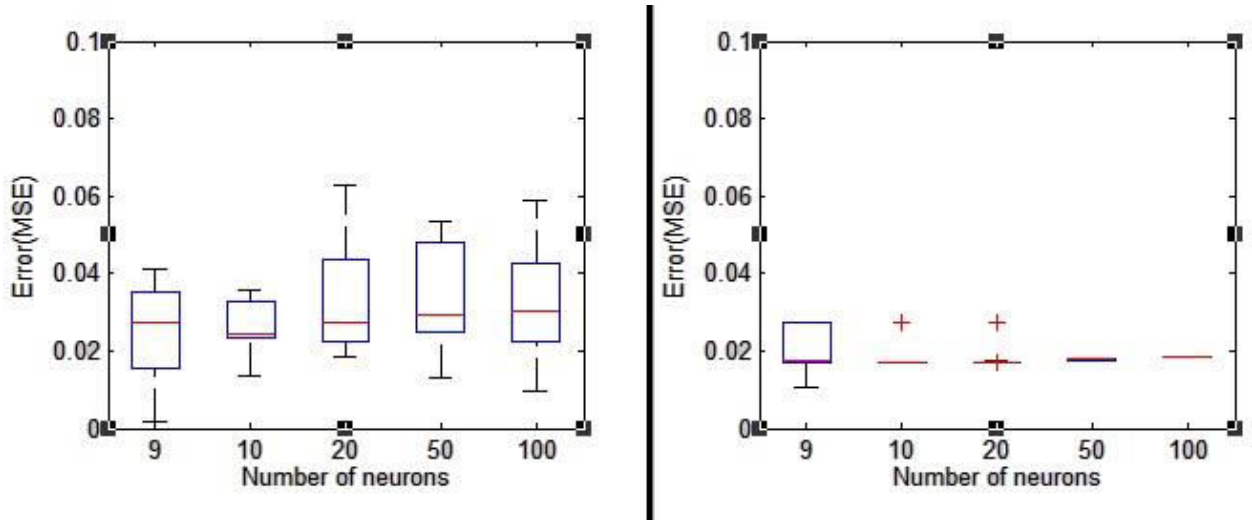
Figure 5.3 is a box plot for the iris dataset. It can be seen from the figures (i) and (ii) that the MSE obtained is more for back propagation algorithm as compared to the MSE computed by using proposed algorithm. The range variation is also more and vast in case of back propagation. The only similarity between two box plots is that the MSE increases with gradual increase of number of neurons for this dataset. But we can't conclude the same for all input datasets.

Figure 5.4 shows the box plot of MSEs calculated when network trained with back propagation learning algorithm and with proposed algorithm for breast cancer dataset. For back propagation algorithm, the average MSE remains the same whereas the variation of maximum and minimum values for a particular number of neuron is large. Similarly for proposed algorithm, the average MSE remains the same but the variation of minimum to maximum value is quite small for different number of hidden layer neurons.



(i) (ii)

Figure 5.3 Box Plot of Iris Dataset MSE vs. Number of Neurons When Trained Using (i) Back Propagation Algorithm (ii) Proposed Algorithm

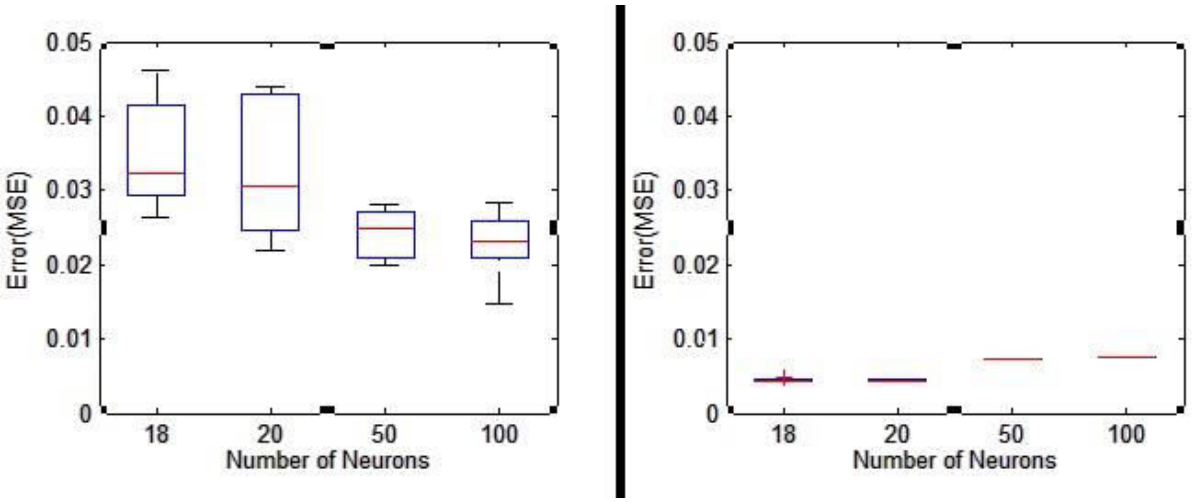


(i) (ii)

Figure 5.4 Box Plot of Breast Cancer Dataset MSE vs. Number of Neurons When Trained Using (i) Back Propagation Algorithm (ii) Proposed Algorithm

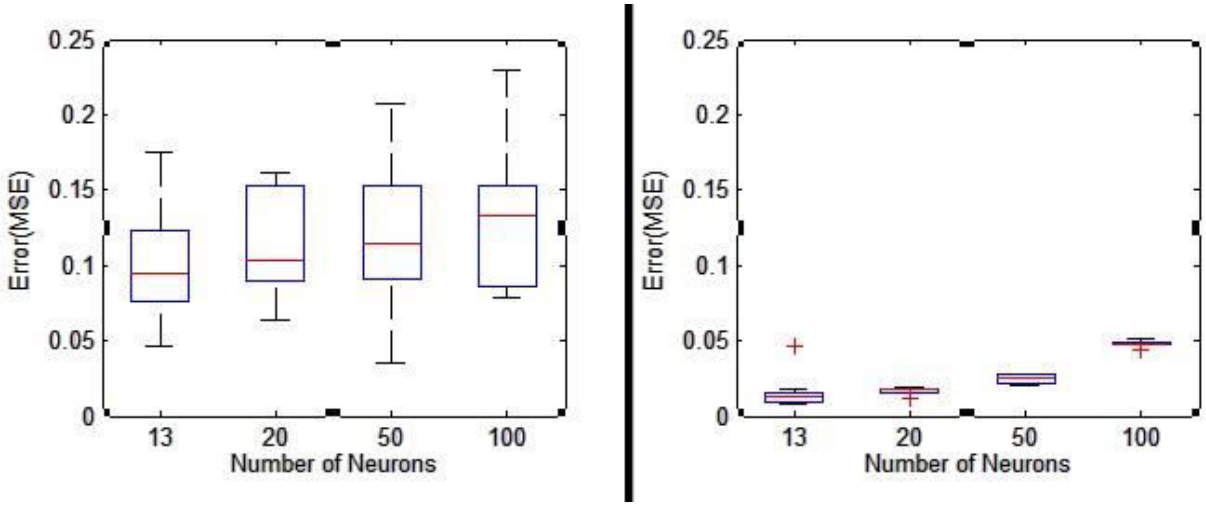
Figure 5.5 and figure 5.6 represents the MSE box plots for the gesture phase and heart disease datasets. Using the back propagation, the average MSE for both the datasets is more than the average MSE obtained using the proposed algorithm. The heart disease dataset follows the pattern of increasing MSE using both algorithms with increase in number of neurons whereas

in case of gesture phase it decreases for back propagation and slightly increases in proposed algorithm case with increase in the number of neuron.



(i) (ii)

Figure 5.5 Box Plot of Gesture Phase Dataset MSE vs. Number of Neurons When Trained Using (i) Back Propagation Algorithm (ii) Proposed Algorithm



(i) (ii)

Figure 5.6 Box Plot of Heart Disease Dataset MSE vs. Number of Neurons When Trained Using (i) Back Propagation Algorithm (ii) Proposed Algorithm

For other three datasets, that is breast cancer, heart disease and gesture phase, the MSE computed using proposed algorithm increases with increase in the number of neurons but the same theory is not applicable in case of the MSE calculated using back-propagation. The

other research regarding MSE remains the same as of the iris dataset. As the variation between the minimum value of MSE to maximum value for proposed algorithm is very less, it can be inferred that the system becomes more stable as compared to when it is trained with back propagation algorithm.

CHAPTER 6

CONCLUSION AND FUTURE PROSPECTS

6.1 CONCLUDING REMARKS

With the increasing demand of machine learning and artificial intelligence in technical world and real time applications, new learning rules and algorithms are being proposed and implemented. These new algorithms must be less time consuming and should have high efficiency in order to be applicable to new technologies. Every other prominent firm and company be it electronics or IT industry is practicing machine learning for different purposes. The most usual application and implementation can be seen at social media and online shopping level. So, the companies like Microsoft, Google, Apple are exploring the field of artificial intelligence along with machine learning. The other fields like finance, security, health are also susceptible to machine learning.

The research work in this thesis presents an algorithm of machine learning implementing neural networks. The hidden layer employs Hebbian-LMS algorithm whereas the output layer is trained with LMS algorithm. The proposed algorithm results are comparable to the existing algorithms. Its advantage over Hebbian is that the weights not only keeps increasing, it can increase or decrease depending upon the neuron state which provides it more stability than Hebb's rule. This algorithm not only specifies the direction of change of weight but also specifies the rate of change. Hebbian-LMS alone is an unsupervised learning but when combines with LMS in neural network, the whole proposed algorithm becomes supervised. The mean square error (MSE) analyzed for the proposed algorithm found to be less than the normal back propagation algorithm MSE, and the range of its maximum value and minimum value was quite less.

6.2 FUTURE PROSPECTS

The research work proposed in this thesis presents a machine learning algorithm which shows better or comparable results to the existing algorithm. It has tendency to reduce the error and hence making it easy to learn from and fast algorithm. Its implementation to artificial neural network makes it more viable to numerous practical engineering applications like signal processing, control circuits, pattern recognition and classification, and even in forensic field

of science. Along with engineering applications, this algorithm can be applied in the field of biology. The neurobiologist are trying to figure out that what kind of learning takes place in human brain or in neural system of living organisms. And by studying and analyzing the Hebbian-LMS learning algorithm in more detail, it is expected that some part of brain must have been performing this learning. Hebbian-LMS extension with other algorithms just as in this research work with normal LMS algorithm give insight to its more vast features and characteristics. The future scope of Hebbian-LMS makes scientists and engineers to look into its other extensions. It is believed if extended in right direction, more accurate results can be obtained. And if this is the learning which takes place in human nervous system, the psychobiology field will has better chance to treat patients with psychiatric problems.

REFRENCES

[1] W.S. McCulloch, W. Pitts, "A logical calculus of ideas immanent in nervous activity", *Bulletin of Mathematical Biophysics*, Vol. 5, pp. 115-133, 1943.

[2] D. O. Hebb, *The Organization of Behavior*, Wiley, New York, 1949.

[3] B. Widrow, M. E. Hoff Jr., "Adaptive switching circuits," in *Proc. IRE WESCON Convention Rec.*, Vol. 4 pp. 96–104, 1960.

[4] B. Widrow, "Bootstrap Learning in Threshold Logic Systems," *Proceedings of the 3rd International Conference of the IFAC*, 1966.

[5] B. Widrow, S. D. Stearns, *Adaptive Signal Processing*. Englewood Cliffs, NJ: Prentice Hall, 1985.

[6] T. Kohonen, *Self-organization and Associative Memory*, Springer Verlag, Berlin, 1988.

[7] Zhi-Quan Luo, "On the convergence of the LMS algorithm with adaptive learning rate for linear feed-forward networks", *Neural Computation*, Vol. 3 Issue 2, pp. 226 - 245, 1991.

[8] M. Riedmiller, H. Braun, "A direct adaptive method for faster back-propagation learning", *Proc. IEEE int. Conf. on Neural Networks*, San Francisco, 1993.

[9] J. Zupan, J. Gasteiger, *Neural Networks for Chemists*, VCH, New York, 1993.

[10] S. Haykin, *Neural Networks - A Comprehensive Foundation*, Macmillan, New York 1994.

[11] C.M. Bishop, *Neural Networks for Pattern Recognition*, Oxford Univ. Press, Oxford, 1995.

[12] R. Rojas, *Neural Networks: A Systematic Introduction*, Springer, Berlin, 1996.

[13] B.D. Ripley, *Pattern Recognition and Neural Networks*, Cambridge Univ. Press, Cambridge, 1996.

[14] Daniel Svozil, Vladimir Kvasnicka, Jiri Pospichal," Introduction to multi-layer feed-forward neural networks", *Chemometrics and Intelligent Laboratory Systems*, Vol. 39, pp. 43-62, 1997.

[15] Yuko Munakata, Jason Pfaffly, "Hebbian learning and development", *Developmental Science*, Vol. 7, Issue 2, pp. 141–148, 2004.

[16] Zoubin Ghahramani, "Unsupervised Learning", *Advanced Lectures on Machine Learning LNAI 3176*, pp. 72–112, Springer-Verlag, 2004.

[17] Jennifer G. Dy, Carla E. Brodley, "Feature Selection for Unsupervised Learning", *Journal of Machine Learning Research*, Vol. 5, pp. 845–889, 2004.

[18] M.Muehlenbrock, *et al.*, "Learning to Detect User Activity and Availability from a Variety of Sensor Data", *IEEE International Conference on Pervasive Computing and Communications*, pp. 13-23, 2004.

[19] Oliver Brdiczka, Patrick Reignier, James L. Crowley, "Supervised Learning of an Abstract Context Model for an Intelligent Environment" , *Joint sOc-EUSAI conference, Grenoble*, 2005.

[20] E. Acar, B.Yener,"Unsupervised multiway data analysis: a literature survey" *IEEE Trans. Knowledge Data Eng,* Vol. 21, pp. 6–20, 2009.

[21] P. Brodka, P. Stawiak, P. Kazienko, "Shortest path discovery in the multi-layered social network", *International Conference on Advances in Social Networks Analysis and Mining (ASONAM),IEEE,* pp. 497–501, 2011.

[22] Vaibhav Narula *et al.*, "Assessment of variants of LMS algorithms for noise cancellation in low and medium frequency signals", *Recent Advancements in Electrical, Electronics and Control Engineering (ICONRAEeCE), International Conference*, 2011.

[23] Bernard Widrow, Aaron Greenblatt, Youngsik Kim, Dookun Park, “The No-Prop algorithm: A new learning algorithm for multilayer neural networks,” *Neural Network*, Vol. 37, pp. 182-188, Jan. 2012.

[24] B. Widrow, J. C. Aragon, “Cognitive memory,” *Neural Network.*, Vol. 41, pp. 3–14, 2013.

[25] R. Sathya, A. Abraham, "Comparison of Supervised and Unsupervised Learning Algorithms for Pattern Classification", *(IJARAI) International Journal of Advanced Research in Artificial Intelligence*, Vol. 2, No. 2, 2013.

[26] Y. Choe, “Anti-Hebbian learning,” *Encyclopedia of Computational Neuroscience*, Berlin, Germany: Springer-Verlag, pp. 1–4, 2014.

[27] Mikko Kivela *et al.* , "Multilayer networks", *Journal of Complex Networks,* Vol. 2, pp. 203-271, 2014.

[28] Bernard Widrow, Youngsik Kim, Dookun Park, "The Hebbian-LMS Learning", *IEEE Computational Intelligence Magazine,* Vol. 10, Issue 4, pp. 37 - 53, Nov. 2015.

[29] M. Lichman. (2013). UCI machine learning repository. [Online]. Available: http://archive ics.uci.edu/ml.

[30] Hoon Chung, Sung Joo Lee, Jeon Gue Park, "Deep neural network using trainable activation functions", *Neural Networks (IJCNN), IEEE International Joint Conference*, November 2016.

LIST OF PUBLICATIONS

1. Vartika, Sakshi "Implementation of Hebbian-LMS learning algorithm" communicated in "*3rd international conference on Next Generation Computing Technologies*", Proceedings in Springer's CCIS series indexed in Scopus.

Vartika_12-07-2017

by Vartika Chauhan

FILE	CHAPTER_1.DOC (797K)		
TIME SUBMITTED	12-JUL-2017 12:44PM	WORD COUNT	10142
SUBMISSION ID	830399556	CHARACTER COUNT	54415

Vartika_12-07-2017

ORIGINALITY REPORT

%16	%6	%13	%7
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Widrow, Bernard, Youngsik Kim, and Dookun Park. "The Hebbian-LMS Learning Algorithm", IEEE Computational Intelligence Magazine, 2015. Publication	%1
2	thesai.org Internet Source	%1
3	Submitted to Higher Education Commission Pakistan Student Paper	%1
4	Yuko Munakata. "Hebbian learning and development", Developmental Science, 4/2004 Publication	%1
5	Zhi-Quan Luo. "On the Convergence of the LMS Algorithm with Adaptive Learning Rate for Linear Feedforward Networks", Neural Computation, 06/1991 Publication	%1
6	Submitted to CSU, San Jose State University Student Paper	%1

7	www.igi-global.com Internet Source	%1
8	www.e-grid.net Internet Source	%1
9	academic.odysci.com Internet Source	%1
10	eprints.pascal-network.org Internet Source	%1
11	Widrow, Bernard, and Juan Carlos Aragon. "Cognitive memory", Neural Networks, 2013. Publication	<%1
12	Widrow, Bernard, Aaron Greenblatt, Youngsik Kim, and Dookun Park. "The No-Prop algorithm: A new learning algorithm for multilayer neural networks", Neural Networks, 2013. Publication	<%1
13	Mohammed, . "Signature Generation Algorithms for Polymorphic Worms", Automatic Defense Against Zero-day Polymorphic Worms in Communication Networks, 2013. Publication	<%1
14	www.ijcset.com Internet Source	<%1
15	Submitted to Thapar University, Patiala Student Paper	<%1

16	Submitted to Malaviya National Institute of Technology Student Paper	<%1
17	Methods in Molecular Biology, 2015. Publication	<%1
18	A., Mohammed. "Neural Network and Adaptive Neuro-Fuzzy Inference System Applied to Civil Engineering Problems", Fuzzy Inference System - Theory and Applications, 2012. Publication	<%1
19	Submitted to University of Witwatersrand Student Paper	<%1
20	Wang, Boxin Man, Teng Jin, Henan. "Prediction of expansion behavior of self-stressing concrete by artificial neural networks and fuzzy ", Construction and Building Materials, June 1 2015 Issue Publication	<%1
21	Huguet, S.. "Use of acoustic emission to identify damage modes in glass fibre reinforced polyester", Composites Science and Technology, 200208 Publication	<%1
22	Submitted to University of Hong Kong Student Paper	<%1

23	James L. Crowley. "Supervised learning of an abstract context model for an intelligent environment", Proceedings of the 2005 joint conference on Smart objects and ambient intelligence innovative context-aware services usages and technologies - sOc-EUSAI 05 sOc-EUSAI 05, 2005 Publication	<%1
24	Shi, Yufeng, Yan Wang, Junru Cao, Shan Zhong, and Jiaolong Wei. "", Second International Conference on Space Information Technology, 2007. Publication	<%1
25	Submitted to Deakin University Student Paper	<%1
26	Submitted to St. Patrick's College Student Paper	<%1
27	ijera.com Internet Source	<%1
28	Submitted to Royal Holloway and Bedford New College Student Paper	<%1
29	Submitted to American Public University System Student Paper	<%1
	achers-ben-johnsonfiercebiotech.com	

30	Internet Source	<%1
31	Marjan Korosec. "Technological information extraction of free form surfaces using neural networks", Neural Computing and Applications, 05/21/2007 Publication	<%1
32	www.ijaiem.org Internet Source	<%1
33	mro.massey.ac.nz Internet Source	<%1
34	Submitted to University of Wales central institutions Student Paper	<%1
35	Submitted to Athlone Institute of Technology Student Paper	<%1
36	Submitted to University of Oxford Student Paper	<%1
37	Ji, Li-Qun. "An assessment of agricultural residue resources for liquid biofuel production in China", Renewable and Sustainable Energy Reviews, 2015. Publication	<%1
38	Submitted to City University Student Paper	<%1

39	Hussain, Amir, Dacheng Tao, Jonathan Wu, and Dongbin Zhao. "Computational Intelligence for Changing Environments [Guest Editorial]", IEEE Computational Intelligence Magazine, 2015. Publication	<%1
40	www.usecasemaps.org Internet Source	<%1
41	Shi, . "Neural Computation", Intelligence Science, 2012. Publication	<%1
42	Cichocki. "Blind Decorrelation and SOS for Robust Blind Identification", Adaptive Blind Signal and Image Processing, 05/02/2002 Publication	<%1
43	Submitted to Queen Mary and Westfield College Student Paper	<%1
44	research.ijcaonline.org Internet Source	<%1
45	A. Stafylopatis. "Pictorial information retrieval using the random neural network", IEEE Transactions on Software Engineering, 07/1992 Publication	<%1
46	Xu, Baoyu Zhang, Hongjun Wang, Zhiteng W. "Model and algorithm of BP neural network	<%1

based on expanded multichain quantum optimization.(Research", Mathematical Problems in Engineering, Annual 2015 Issue
Publication

---

47 Darsey, Jerry A., William O. Griffin, Sravanthi Joginipelli, and Venkata Kiran Melapu. "Architecture and Biological Applications of Artificial Neural Networks: A Tuberculosis Perspective", Methods in Molecular Biology, 2015.
Publication

<%1

---

48 Mohammed Bahoura. "FPGA-Implementation of Parallel and Sequential Architectures for Adaptive Noise Cancelation", Circuits Systems and Signal Processing, 05/07/2011
Publication

<%1

---

49 arizona.openrepository.com
Internet Source

<%1

---

50 XU, Jie, Keiji YAMADA, Katsuhiko SEIKIYA, Ryutaro TANAKA, and Yasuo YAMANE. "Comparison of applying static and dynamic features for drill wear prediction", Journal of Advanced Mechanical Design Systems and Manufacturing, 2014.
Publication

<%1

---

51 Sathya, R., and Annamma Abraham. "Comparison of Supervised and Unsupervised

<%1

Learning Algorithms for Pattern Classification", INTERNATIONAL JOURNAL OF ADVANCED RESEARCH IN ARTIFICIAL INTELLIGENCE, 2013.
Publication

---

52 Lecture Notes in Computer Science, 2004. <%1
Publication

---

53 Moreira Sá de Souza, Luciana. "Wireless Sensor Network Pattern Based Fault Isolation in Industrial Applications", Universität Karlsruhe, 2009. <%1
Publication

EXCLUDE QUOTES ON
EXCLUDE BIBLIOGRAPHY ON
EXCLUDE MATCHES < 5 WORDS