MEMORY STORAGE IN A VARIABLE THRESHOLD ARTIFICIAL NEURAL NETWORK

By

PRAVEEN KURUVADA

Bachelor of Technology in Information Technology

Jawaharlal Nehru Technological University

Hyderabad, Andhra Pradesh

2009

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
July, 2011

MEMORY STORAGE IN A VARIABLE THRESHOLD

ARTIFICIAL NEURAL NETWORK

Thesis Approved:

Dr. Subhash Kak

Thesis Adviser

Dr. Johnson Thomas

Dr. Michel Toulouse

Dr. Mark E. Payton

Dean of the Graduate College

ACKNOWLEDGMENTS

It gives me immense pleasure in acknowledging all the people, without whom, this thesis would have been impossible.

I owe deep gratitude and respect to my thesis advisor Dr. Subhash Kak who has always guided me through my thesis work. It was his support, enthusiasm and the patience with which he guided me and appreciated my ideas that promoted me to further explore over this topic. It was his timely advice, review of work and guidance that led me this far. This master's thesis would not have been possible without his support.

I would like to take this opportunity to express my gratitude and thanks to Dr. David R Porter for supporting me through my study at OSU.I have worked with a great team in solving the departmental software issues. I have learned a lot working under him and would be always thankful to him. I would like to thank Dr. Thomas under whom I learned how to deal with databases. I would also like to thank Dr. Michel Toulouse for his invaluable comments and support during my thesis.

Lastly and most importantly I would like to dedicate this thesis to my parents KVSS Prasad and K Sujatha and My Sister Sirisha Nandyala and my brother in-law Pavan Nandyala for believing in me and giving me constant support in everything I intended to do.

TABLE OF CONTENTS

Chapter Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

CHAPTER I

INTRODUCTION

History of Neural Network

The importance of neural networks as a subject is partly owing to the possibility of the comparison of machines to biological brains. Warren McCulloch and Walter Pitts proposed the first model of a neural network over sixty years ago. The neural network acts as an information-processing unit where the information is received from the organs of the body, which are reacting to the external conditions of the environment. An appropriate response is generated by decoding and processing the information received.

A biological neuron in the human brain collects signals from other neurons through fine structure called dendrites. Each neuron sends out spikes of electrical activity through a long, thin stand known as axon, which splits into thousands of branches. At the end of each of these branches from a axon, a structure called a synapse converts the activity from the axon into electrical effects that excite activity in the connected neurons. When a neuron receives this 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. The actual information processing abilities of a neuron are yet unknown.

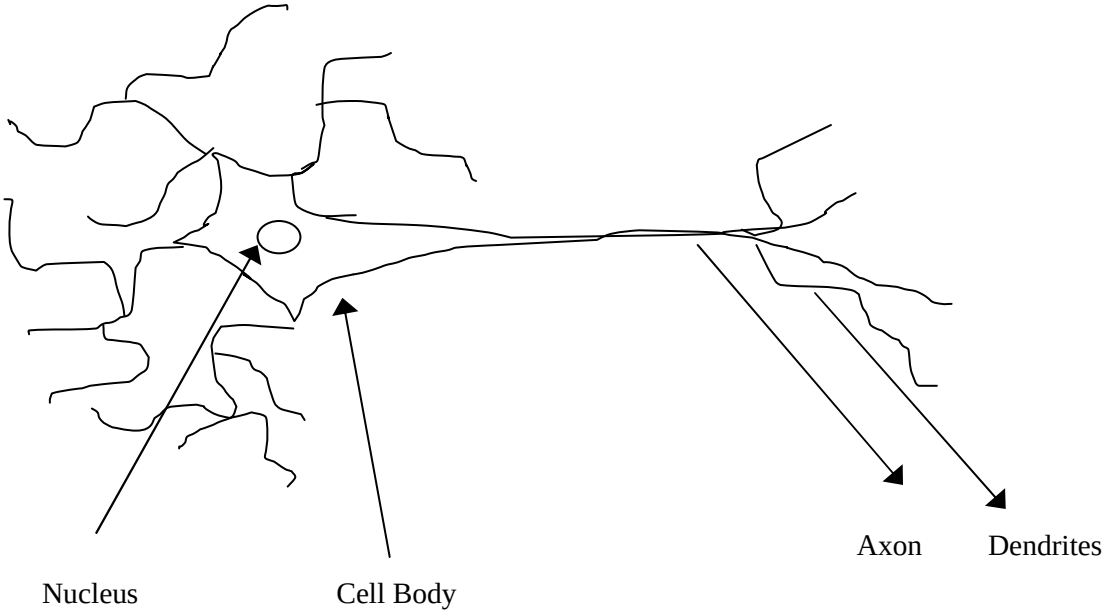

Figure 1. A Biological Neuron

Biological Neurons to Artificial Neurons

An artificial neural network can be constructed by deducing the essential features of neurons and their interconnections. However because our knowledge of neurons is incomplete, our models are necessarily gross simplification of biological neural network.

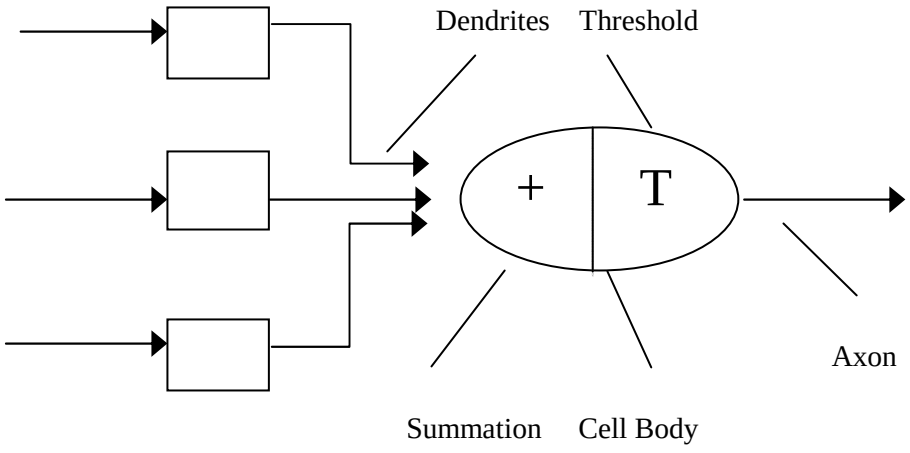

Figure 2. Artificial Neural Network

Architecture of neural networks:

Artificial neural network are of two types:

A **Feedforward** ANN (figure 3) allows signals to travel in only one direction i.e. input to output. There are no loops in the network. Feedforward networks simply associate inputs with outputs. They are largely used in pattern recognition.

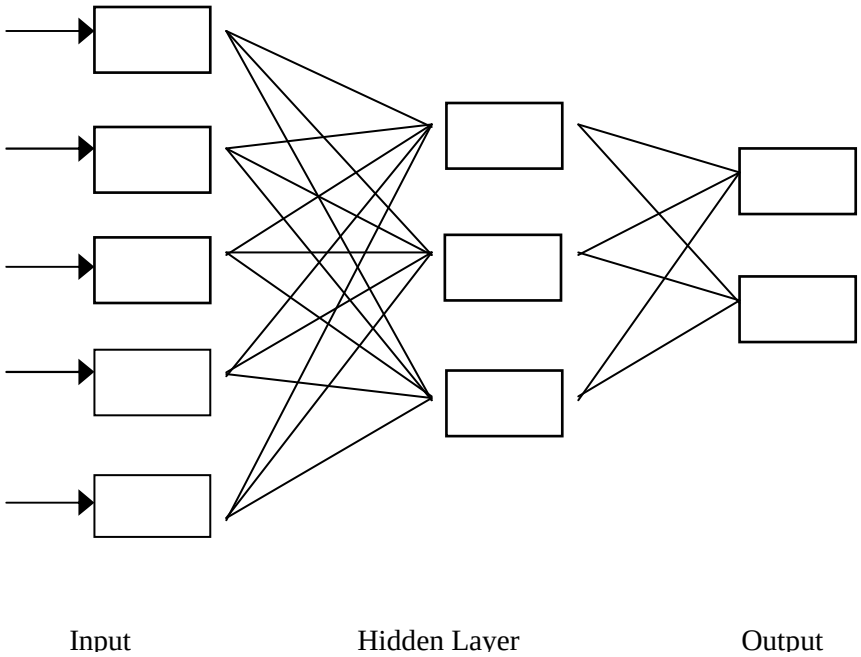

Figure 3. Feed forward neural network

Feedback networks

Feedback networks (figure 4) can have loops in the network i.e. signals can travel in both forward and backward directions. Feedback networks are dynamic and their state changes continuously until they reach an equilibrium point. They remain at the equilibrium point until a change in the input occurs and a new equilibrium point is to be found.

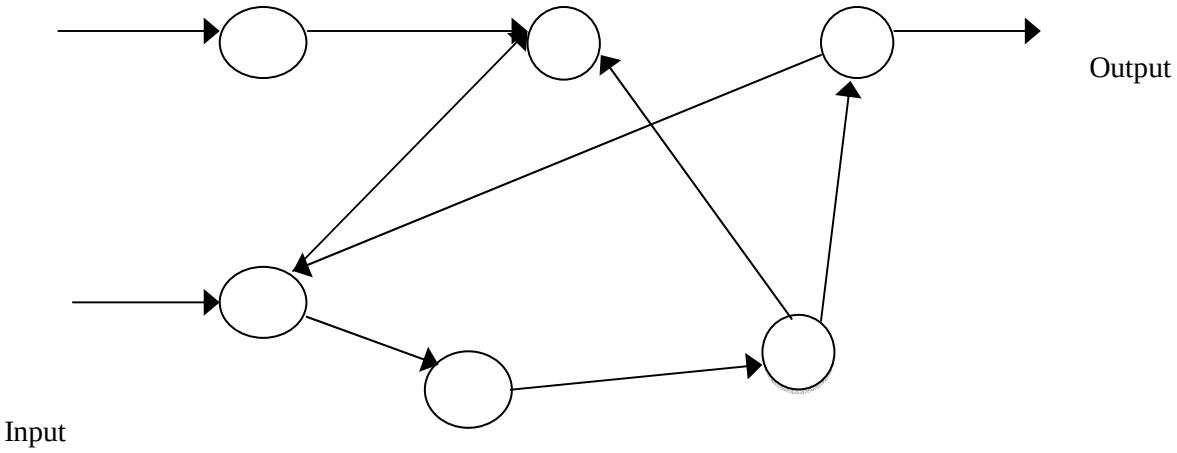

Figure 4. Feedback neural network

Learning in a Neural Network:

The choice of the learning algorithm is critical in defining the properties of a neural network. Learning makes it possible for a processing unit to change its input/output behavior as a result of change in the environment. When the network is constructed the activation rule is usually fixed. To change the input/output behavior the weights corresponding to the input vector must be adjusted. The learning algorithm for training makes it possible for the network to change the input/output behavior. Some of the learning algorithms considered in this thesis are:

Hebbian Model

A model that assumes that the synaptic strength between any two neurons is changed based on the pre-synaptic neuron's persistent stimulation of the post-synaptic neuron was proposed by Donald Hebb in 1949. This model is now known as Hebbian model. The learning rule in the Hebbian model is given by

$$W_{i,j} = x_i . x_j'$$

where x is the input vector and W is a weight n x n matrix containing the weights of successively firing neurons. A memory is considered to be stored in the network if

$$x = sgn(W.x)$$

where sgn is the signum function. The signum function is sgn(k) equals 1 when k is equal to or greater than zero and -1 for rest of the values.

Widrow-Hoff Learning:

A model that was proposed to increase the memory storage capacity of the Hebbian network is the Widrow-Hoff learning rule. In this model the weights of the neurons that are stored in the network are adjusted to increase the possibility of retrieving those memories from the network. When new memories are brought into the network, the learning process of these new memories would have an overwriting effect on the previously stored memories.

Initially the calculation of the error associated with the retrieval of memory from the network is done. Based on this error matrix obtained, the weights are adjusted such that the error associated with the particular memory is minimized. This process is continued until all the memories are stored in the network with minimal error or with a permissible threshold.

Summarizing,

$W_{n+1} = W_n + \Delta(\mathrm{W_n})$, where $\Delta(\mathrm{W_n}) = \eta(\mathrm{x_i} - \mathrm{W_n x_i})$, W is the weight matrix, $\mathrm{x_i}$ is the present input, and η is a small positive constant.

This method of learning based on the error calculation is called the “batch learning” as opposed to “single stimulus learning” of the Hebbian learning. It has been shown that batch learning converges faster to the correct solution when compared to single stimulus learning.

An error vector is estimated for each iteration of weighted matrix adjustment. Then an error term associated with these vectors is calculated and average this error term over the number of memories that are trained to the network. Defining a good error term is one of the major problems of this model.

Information Storage

Information stored in a neural network is generally viewed as a pattern of activities in some models while in others, it is seen as information stored at specific location. Experiments have shown that a binary neural network has a capacity of approximately 0.14N, where N is the number of neurons in the network. A binary neural network barely describes the complexity of a biological neural network, since a actual biological neuron not only carries electrical impulses but it is also associated with a train of voltage spikes. A non-binary neural network has a much higher capacity than that of a binary neural network but the computation complexity in dealing with these networks is high.

In this thesis we describe a learning approach used to store memories in a network using variable thresholds within Hebbian model. This approach determines the threshold corresponding to each neuron in a network. The experimental results of using this approach over different size of networks have been described. This approach has been further applied to the B-Matrix approach of memory retrieval. The capacity and complexities of a non-binary neural network have been discussed.

CHAPTER II

REVIEW OF LITERATURE

B-Matrix Approach

The B-Matrix Approach [12], developed by Kak is a feedback network with indexed memory retrieval and it may be considered as a generator model for memory retrieval. In this the fragment length increases as the activity starts from one neuron and spreads to adjacent neurons. The fragment generated as a result is fed back to the network recursively until the memory is retrieved. The use of proximity matrix along with the B-Matrix approach was shown by Kak [12].

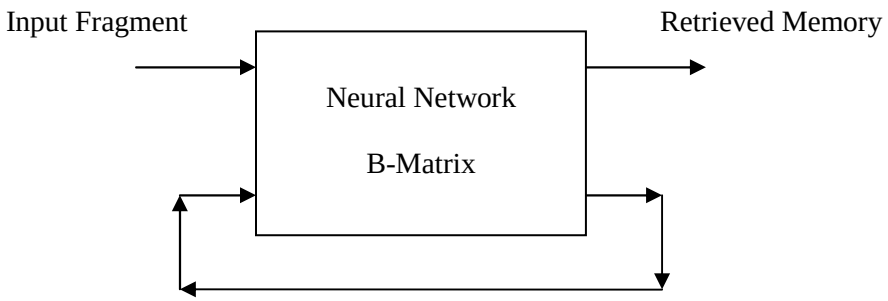

Figure 5. B-Matrix generator model

Recollection of memories in the B-Matrix Approach is by using the lower triangular matrix *B*, constructed as,

$T = B + B^t$

The proximity matrix stores the synaptic strength between the neurons. The activity starts from one neuron and spreads to additional neurons based on these synaptic strengths determined by the proximity matrix. Starting with the fragment f^{1}, the updating proceeds as:

$$f^{i} = \mathrm{sgn}\,(B \cdot f^{i-1}),$$

Where f^{i} is the i^{th} iteration of the generator model. The i^{th} iteration does not alter the i-1 iteration values but only produces the value of the i^{th} binary index of the memory vector.

This model relies heavily on the geometric organization of the network .The proximity matrix provides this information of the geometric proximity of each neuron from every other neuron.

The neural network of *n* neurons may be thought of as a three dimensional network of *n* nodes interconnected with each other with different synaptic strength between each node. We can construct a two dimensional graph of the network as a polygon of *n* sides with all diagonals connected and each corner being a node. For example, consider the neural network of 6 nodes as shown in Figure 6.

Let us assume without loss of generality that this network is already trained with a set of memories. When retrieving a memory from this network, the activity starts from one node and then spreads to the adjacent nodes as described by the proximity matrix. Assume that the activity starts at the second neuron and spreads from there on. If the synaptic order given by the proximity matrix is [2 5 3 1 4 6], then memory retrieval proceeds as shown in the figure.

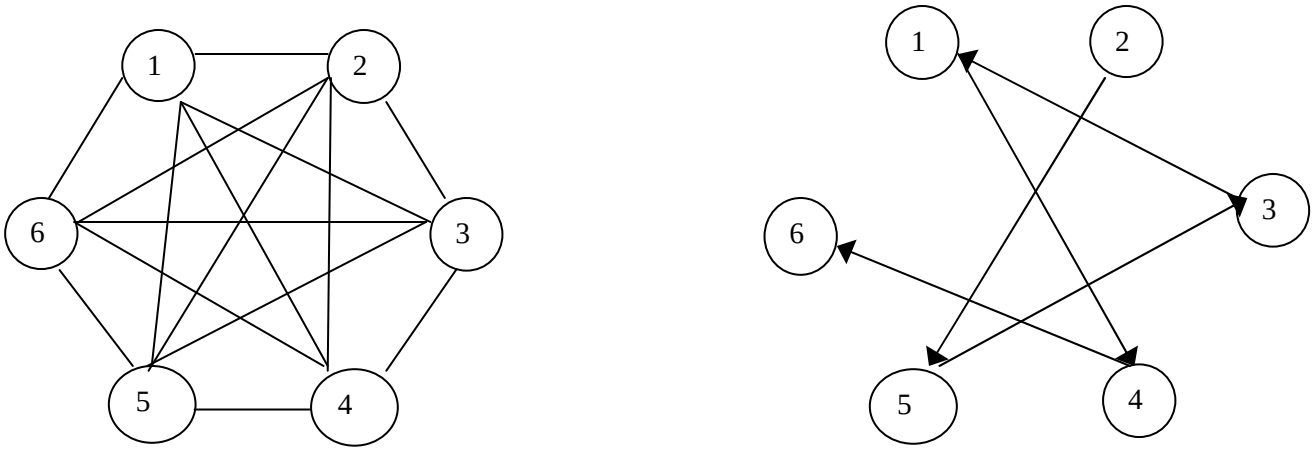

Figure 6. Network diagram for six neurons and the order in which activity spreads in the network based on the proximity matrix.

Each neuron is capable of storing and retrieving a particular memory by the spread of activity as long as the network is not overwhelmed with information. As seen in the above example, it is possible to retrieve the right memories if we do know what the index of the neuron to be stimulated is, and what should be used to stimulate that selected neuron. Hence indexing plays a major role in the retrieval of memories. To better understand how a complex neural network might function, we introduce the concept of a sub-neural net and an indexing neural net. A different approach to indexing is provided in [10] , [15] .

This model eliminates the need for a complete vector of memory needed for verification and a partial fragment of memory would be sufficient for recall. To illustrate this with an example, consider a person listening to a song. The next time he hears the song, he need not listen to the whole song to remember that this was the song he heard. All he needs is a small fragment of the song to help him recollect and maybe even sing the song.

CHAPTER III

METHODOLOGY

Neuroscience Background:

Actual neural networks of the brain are far more complex than the models discussed in the literature and the processing abilities of each neuron is yet unknown. The basic neural network has a neural weight, which represents the strength of the connection for each synapse and a nonlinear threshold in the neuron soma (cell body). As the sum of the weighted inputs to the neuron soma increases there is relatively little response from the neuron until a critical threshold is reached by the neuron, at this point the neuron rapidly increases the output of its axon and fires. Hence different neurons have different threshold.

We assume that varying threshold values are learnt by neurons as part of the training process. The advantages of doing so will be shown both for the standard Hopfield model as well as for the B-matrix model [16],[17].

Variable Threshold Learning Approach

In the variable threshold learning approach each neuron is assumed to have a different threshold, which, as we have argued, is likely to be more realistic than the uniform threshold neuron model. The threshold associated with each neuron is calculated in an iterative manner by increasing the threshold by 0.1 till the maximum numbers of neurons are stored in the network.

Variable Threshold Learning Algorithm:

1) Identify the number of neurons in the network and the number of memories to be stored.
2) Find the T-Matrix associated with the memory vectors considered. The interconnection T-Matrix is constructed by multiplying the memory vectors with their transpose.

$$T = \Sigma\, x^{(i)} x^{(i)t}$$

where the memories are binary column vectors (x^i), composed of {-1, 1} and the diagonal terms are taken to be zero.

3) Find the threshold corresponding to each neuron in the network. This is done in an iterative manner where the threshold at each neuron is increase by a factor of 0.1 and the number of memories that could be stored is determined. This process is continued till the thresholds at each neuron is determined with which the maximum number of memories could be stored into the network.
4) In the variable threshold approach a memory is considered to be stored into the network when

$$X^i = T_i(T \,.\, x^{(i)}).$$

5) When new memories are brought into the network the thresholds are to be calculated again such that maximum memories can be stored into the network (repeat step 3).

The variable threshold approach shows an increase in the number of memories store in the network when a smaller network is considered. The model performs better than the fixed threshold approach when a network of 1000 neurons is considered. The percentage of memories stored decreases when a very larger network is considered. One of the main issues in this model is the overhead in calculating the threshold iteratively when new memories are brought into the network.

Consider the example of a network with 7 neurons and 5 memories are to be stored into the network.

Let's assume that the actual memories are the vectors given below:

X^1= 1 1 1 1 -1 1 -1

X^2= 1 -1 -1 1 -1 1 -1

X^3= 1 -1 1 -1 1 -1 1

X^4= -1 1 1 -1 -1 -1 1

X^5= 1 -1 1 1 1 -1 1

The interconnection matrix or the T-matrix is calculated as

$T= \Sigma x^{(i)}x^{(i)t}$

$$\begin{pmatrix} 0 & -3 & 1 & 3 & 1 & 1 & -1 \\ -3 & 0 & 1 & -1 & -3 & 1 & -1 \\ 1 & 1 & 0 & -1 & 1 & -3 & 3 \\ 3 & -1 & -1 & 0 & -1 & 3 & -3 \\ 1 & -3 & 1 & -1 & 0 & -3 & 3 \\ 1 & 1 & -3 & 3 & -3 & 0 & -5 \\ -1 & -1 & 3 & -3 & 3 & -5 & 0 \end{pmatrix}$$

In the fixed threshold approach a memory is considered to be stored in the network if

$x^{(i)} = sig\ (T*x^{(i)})$

Multiplying $T*X_i$ where i=1,2,3,4,5,

$$\begin{pmatrix} 0 & -3 & 1 & 3 & 1 & 1 & -1 \\ -3 & 0 & 1 & -1 & -3 & 1 & -1 \\ 1 & 1 & 0 & -1 & 1 & -3 & 3 \\ 3 & -1 & -1 & 0 & -1 & 3 & -3 \\ 1 & -3 & 1 & -1 & 0 & -3 & 3 \\ 1 & 1 & -3 & 3 & -3 & 0 & -5 \\ -1 & -1 & 3 & -3 & 3 & -5 & 0 \end{pmatrix} * \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ -1 \\ 1 \\ -1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \end{pmatrix}$$

$$\begin{pmatrix} 0 & -3 & 1 & 3 & 1 & 1 & -1 \\ -3 & 0 & 1 & -1 & -3 & 1 & -1 \\ 1 & 1 & 0 & -1 & 1 & -3 & 3 \\ 3 & -1 & -1 & 0 & -1 & 3 & -3 \\ 1 & -3 & 1 & -1 & 0 & -3 & 3 \\ 1 & 1 & -3 & 3 & -3 & 0 & -5 \\ -1 & -1 & 3 & -3 & 3 & -5 & 0 \end{pmatrix} * \begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \end{pmatrix}$$

$$\begin{pmatrix} 0 & -3 & 1 & 3 & 1 & 1 & -1 \\ -3 & 0 & 1 & -1 & -3 & 1 & -1 \\ 1 & 1 & 0 & -1 & 1 & -3 & 3 \\ 3 & -1 & -1 & 0 & -1 & 3 & -3 \\ 1 & -3 & 1 & -1 & 0 & -3 & 3 \\ 1 & 1 & -3 & 3 & -3 & 0 & -5 \\ -1 & -1 & 3 & -3 & 3 & -5 & 0 \end{pmatrix} * \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} 0 & -3 & 1 & 3 & 1 & 1 & -1 \\ -3 & 0 & 1 & -1 & -3 & 1 & -1 \\ 1 & 1 & 0 & -1 & 1 & -3 & 3 \\ 3 & -1 & -1 & 0 & -1 & 3 & -3 \\ 1 & -3 & 1 & -1 & 0 & -3 & 3 \\ 1 & 1 & -3 & 3 & -3 & 0 & -5 \\ -1 & -1 & 3 & -3 & 3 & -5 & 0 \end{pmatrix} * \begin{pmatrix} -1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \\ 1 \end{pmatrix} = \begin{pmatrix} -1 \\ 1 \\ 1 \\ -1 \\ 1 \\ -1 \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} 0 & -3 & 1 & 3 & 1 & 1 & -1 \\ -3 & 0 & 1 & -1 & -3 & 1 & -1 \\ 1 & 1 & 0 & -1 & 1 & -3 & 3 \\ 3 & -1 & -1 & 0 & -1 & 3 & -3 \\ 1 & -3 & 1 & -1 & 0 & -3 & 3 \\ 1 & 1 & -3 & 3 & -3 & 0 & -5 \\ -1 & -1 & 3 & -3 & 3 & -5 & 0 \end{pmatrix} * \begin{pmatrix} 1 \\ -1 \\ 1 \\ 1 \\ 1 \\ -1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \\ 1 \end{pmatrix}$$

From the above we can observe that when the memory vectors are multiplied with the T-Matrix using the fixed threshold approach only one memory could be stored in the network.

The memory stored in the network is

X^3= 1 -1 1 -1 1 -1 1

Applying the variable threshold approach to the above network would require the identification of thresholds corresponding to each neuron through an iterative process. This is done by:

Multiplying each memory vector with the T matrix and determining the corresponding weights.

	N1	N2	N3	N4	N5	N6	N7
X1=	2	2	-6	8	-8	10	-10
X2=	6	0	-8	12	-4	14	-14
X3=	0	-6	8	-4	12	-14	14
X4=	-8	6	6	-10	4	-8	8
X5=	6	-8	6	-4	10	-8	8

From each column the minimum value of weight is initially set as the threshold and the number of memories that could be stored is tested. The threshold is increased by a factor of 0.1 and the number of memories stored in the network is determined. This process continues until the maximum possible memories are stored into the network. For the example considered above the threshold values at each neuron are found to be:

-7.9 is the Threshold at Neuron1

0.1 is the Threshold at Neuron2

-7.9 is the Threshold at Neuron3

-3.9 is the Threshold at Neuron4

4.1 is the Threshold at Neuron5

-7.9 is the Threshold at Neuron6

-9.9 is the Threshold at Neuron7

-7.9

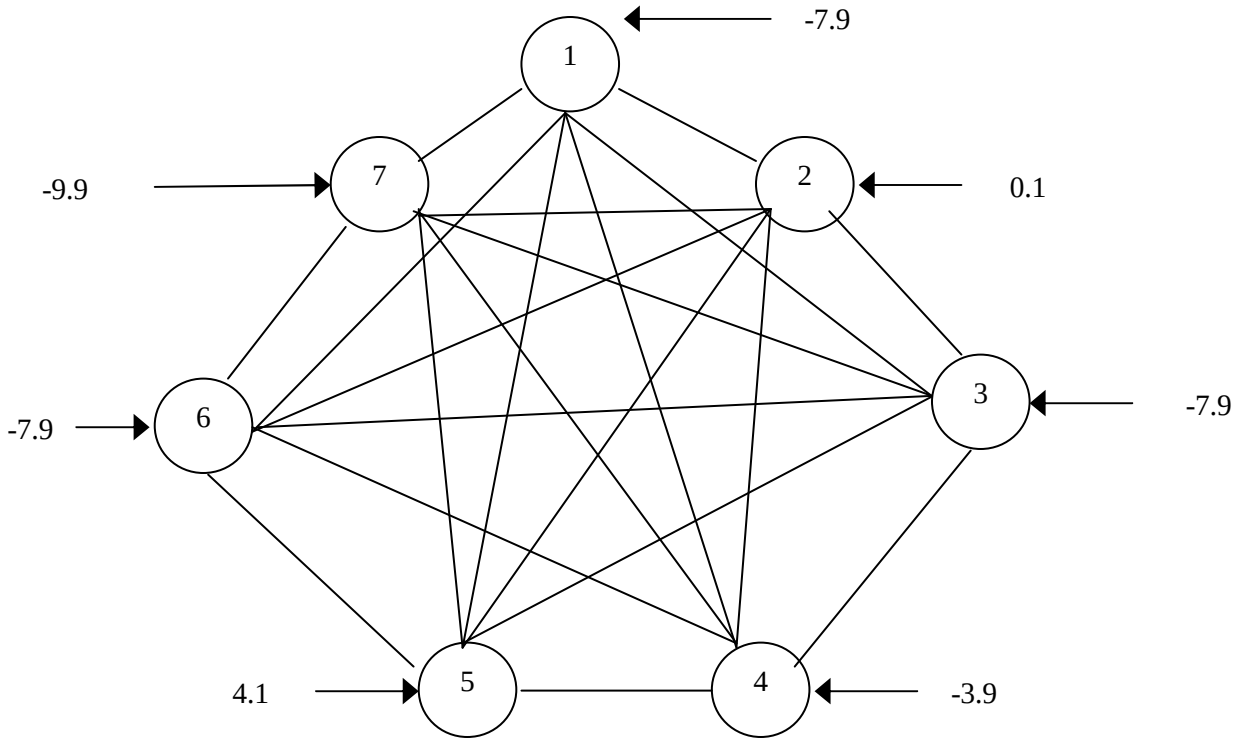

Figure 7. Network for above Example with thresholds at each neuron.

Using these threshold values we determine the number of memories stored in the network:

T-Matrix * X^1 Weighted Sum T_i Output

$$\begin{pmatrix} 0 & -3 & 1 & 3 & 1 & 1 & -1 \\ -3 & 0 & 1 & -1 & -3 & 1 & -1 \\ 1 & 1 & 0 & -1 & 1 & -3 & 3 \\ 3 & -1 & -1 & 0 & -1 & 3 & -3 \\ 1 & -3 & 1 & -1 & 0 & -3 & 3 \\ 1 & 1 & -3 & 3 & -3 & 0 & -5 \\ -1 & -1 & 3 & -3 & 3 & -5 & 0 \end{pmatrix} * \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ -1 \\ 1 \\ -1 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \\ -6 \\ 8 \\ -8 \\ 10 \\ -10 \end{pmatrix} \begin{bmatrix} -7.9 \\ 0.1 \\ -7.9 \\ -3.9 \\ 4.1 \\ -7.9 \\ -9.9 \end{bmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ -1 \\ 1 \\ -1 \end{pmatrix}$$

T-Matrix $* X^2$ | Weighted Sum | T_i | Output

$$\begin{pmatrix} 0 & -3 & 1 & 3 & 1 & 1 & -1 \\ -3 & 0 & 1 & -1 & -3 & 1 & -1 \\ 1 & 1 & 0 & -1 & 1 & -3 & 3 \\ 3 & -1 & -1 & 0 & -1 & 3 & -3 \\ 1 & -3 & 1 & -1 & 0 & -3 & 3 \\ 1 & 1 & -3 & 3 & -3 & 0 & -5 \\ -1 & -1 & 3 & -3 & 3 & -5 & 0 \end{pmatrix} * \begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \end{pmatrix} = \begin{pmatrix} 6 \\ 0 \\ -8 \\ 12 \\ -4 \\ 14 \\ -14 \end{pmatrix} \begin{bmatrix} -7.9 \\ 0.1 \\ -7.9 \\ -3.9 \\ 4.1 \\ -7.9 \\ -9.9 \end{bmatrix} = \begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \end{pmatrix}$$

T-Matrix $* X^3$ | Weighted Sum | T_i | Output

$$\begin{pmatrix} 0 & -3 & 1 & 3 & 1 & 1 & -1 \\ -3 & 0 & 1 & -1 & -3 & 1 & -1 \\ 1 & 1 & 0 & -1 & 1 & -3 & 3 \\ 3 & -1 & -1 & 0 & -1 & 3 & -3 \\ 1 & -3 & 1 & -1 & 0 & -3 & 3 \\ 1 & 1 & -3 & 3 & -3 & 0 & -5 \\ -1 & -1 & 3 & -3 & 3 & -5 & 0 \end{pmatrix} * \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ -6 \\ 8 \\ -4 \\ 12 \\ -14 \\ 14 \end{pmatrix} \begin{bmatrix} -7.9 \\ 0.1 \\ -7.9 \\ -3.9 \\ 4.1 \\ -7.9 \\ -9.9 \end{bmatrix} = \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \\ 1 \end{pmatrix}$$

T-Matrix $* X^4$ | Weighted Sum | T_i | Output

$$\begin{pmatrix} 0 & -3 & 1 & 3 & 1 & 1 & -1 \\ -3 & 0 & 1 & -1 & -3 & 1 & -1 \\ 1 & 1 & 0 & -1 & 1 & -3 & 3 \\ 3 & -1 & -1 & 0 & -1 & 3 & -3 \\ 1 & -3 & 1 & -1 & 0 & -3 & 3 \\ 1 & 1 & -3 & 3 & -3 & 0 & -5 \\ -1 & -1 & 3 & -3 & 3 & -5 & 0 \end{pmatrix} * \begin{pmatrix} -1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \\ 1 \end{pmatrix} = \begin{pmatrix} -8 \\ 6 \\ 6 \\ -10 \\ 4 \\ -8 \\ 8 \end{pmatrix} \begin{bmatrix} -7.9 \\ 0.1 \\ -7.9 \\ -3.9 \\ 4.1 \\ -7.9 \\ -9.9 \end{bmatrix} = \begin{pmatrix} -1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \\ 1 \end{pmatrix}$$

T-Matrix $* X^5$ Weighted Sum T_i Output

$$\begin{pmatrix} 0 & -3 & 1 & 3 & 1 & 1 & -1 \\ -3 & 0 & 1 & -1 & -3 & 1 & -1 \\ 1 & 1 & 0 & -1 & 1 & -3 & 3 \\ 3 & -1 & -1 & 0 & -1 & 3 & -3 \\ 1 & -3 & 1 & -1 & 0 & -3 & 3 \\ 1 & 1 & -3 & 3 & -3 & 0 & -5 \\ -1 & -1 & 3 & -3 & 3 & -5 & 0 \end{pmatrix} * \begin{pmatrix} 1 \\ -1 \\ 1 \\ 1 \\ 1 \\ -1 \\ 1 \end{pmatrix} = \begin{pmatrix} 6 \\ -8 \\ 6 \\ -4 \\ 10 \\ -8 \\ 8 \end{pmatrix} \begin{bmatrix} -7.9 \\ 0.1 \\ -7.9 \\ -3.9 \\ 4.1 \\ -7.9 \\ -9.9 \end{bmatrix} = \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \\ 1 \end{pmatrix}$$

From the above we can observe that by applying the variable threshold approach we could store four memories into the network.

X^1= 1 1 1 1 -1 1 -1

X^2= 1 -1 -1 1 -1 1 -1

X^3= 1 -1 1 -1 1 -1 1

X^4= -1 1 1 -1 -1 -1 1

This method gives the same performance when the threshold is increased by a factor of 0.2 up to 1. The change in the output for greater values is due to the sudden increase in the threshold. When the rate at which threshold at each neuron is increased, the number of iterations required in finding the threshold at each neuron decreases, hence increasing the performance. The table below contains the number of iterations for each threshold increase rate. A graph is plotted for the total number of iterations to determine threshold at each neuron for each threshold increase rate. This graph shows the relation between the number of iterations and the threshold increase rate.

Threshold	Neuron1	Neuron2	Neuron3	Neuron4	Neuron5	Neuron6	Neuron7
0.1	1	81	1	61	121	61	41
0.2	1	41	1	31	61	31	21
0.3	1	27	1	21	41	21	14
0.4	1	21	1	16	31	16	11
0.5	1	17	1	13	25	13	9
1	1	8	1	6	12	6	4

Table 1. Number of iterations at each neuron when variable threshold is used

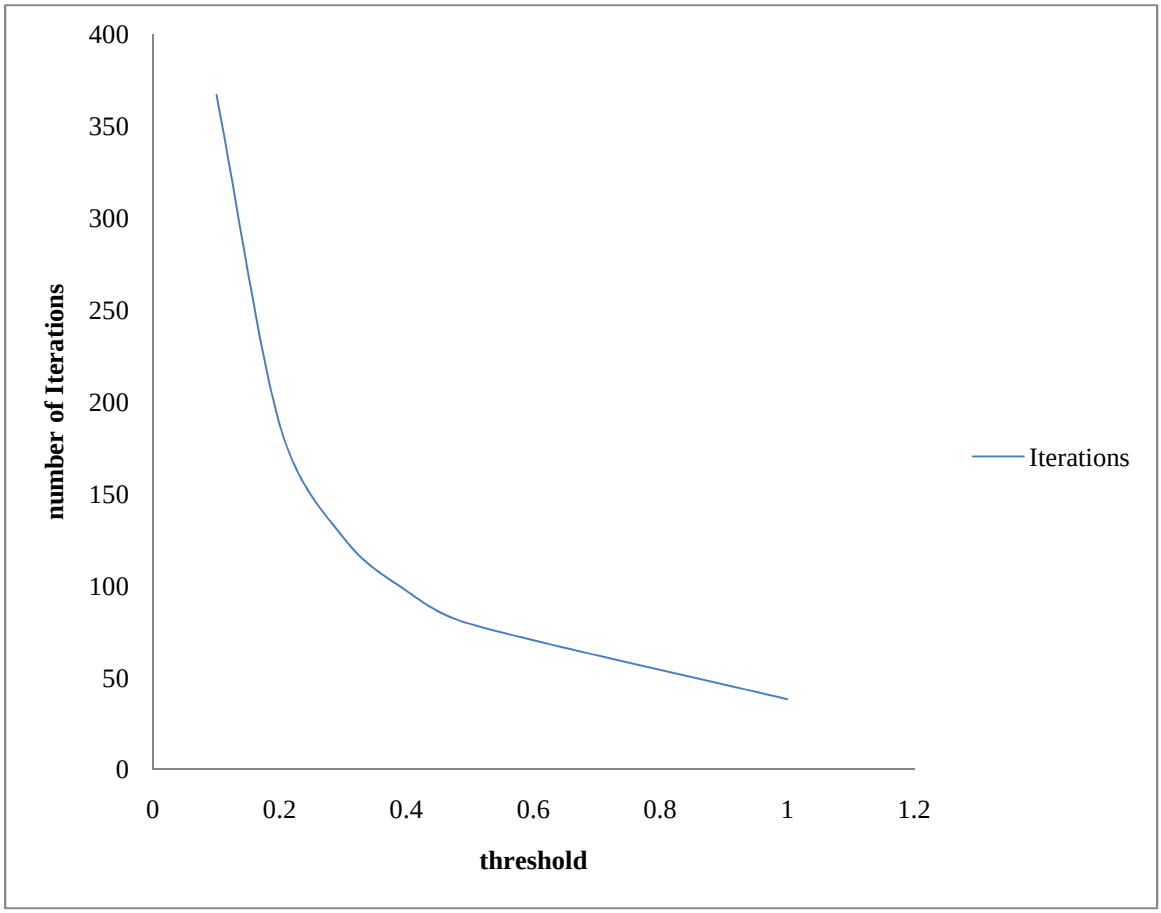

Figure 8. Number of iterations for each threshold increase rate.

This approach is further extended to the B-matrix approach of memory retrieval. In both these cases a Widrow-Hoff technique is used to learn the variable thresholds. The improvement in capacity is substantial.

From a signal-processing implementation point of view, the neural network model is merely a matrix multiplication model and one can view the operations also in a multilayered architecture [18]-[20] that may of relevance in the physical three-dimensional biological structures.

Extending the method to the B-Matrix Approach

Recollection of memories in the B-Matrix approach is by using the lower triangular matrix *B,* constructed as,

$T = B + B^t$

The proximity matrix stores the synaptic strength between the neurons. The activity starts from one neuron and spreads to additional neurons based on these synaptic strengths determined by the proximity matrix. Starting with the fragment f^1, the updating proceeds as:

$f^i = \text{sgn}(B \cdot f^{i-1})$,

Where f^i is the i^{th} iteration of the generator model. The i^{th} iteration does not alter the i-1 iteration values but only produces the value of the i^{th} binary index of the memory vector.

This model relies heavily on the geometric organization of the network .The proximity matrix provides this information of the geometric proximity of each neuron from every other neuron.

Applying the variable threshold approach to the B-matrix requires the use of thresholds corresponding to each neuron during the memory retrieval process (i.e. when the fragment vector is applied on the B-Matrix, the threshold corresponding to specific neuron is considered).

Consider the above example used for storing memories The B-Matrix for the above network is constructed as $T = B + B^t$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 3 & -1 & -1 & 0 & 0 & 0 & 0 \\ 1 & -3 & 1 & -1 & 0 & 0 & 0 \\ 1 & 1 & -3 & 3 & -3 & 0 & 0 \\ -1 & -1 & 3 & -3 & 3 & -5 & 0 \end{pmatrix}$$

To retrieve memories from the network using the B-Matrix we consider a fragment of memory and apply it repeatedly to the network till the entire memory is retrieved. The following example shows how the memories are retrieved from the network starting from a fragment. In the memory retrieval process we consider the variable thresholds that were found during the memory storage process.

B-Matrix * Fragment Vector Weighted Sum T_i Output

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 3 & -1 & -1 & 0 & 0 & 0 & 0 \\ 1 & -3 & 1 & -1 & 0 & 0 & 0 \\ 1 & 1 & -3 & 3 & -3 & 0 & 0 \\ -1 & -1 & 3 & -3 & 3 & -5 & 0 \end{pmatrix} * \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} \\ 2 \\ \\ \\ \\ \\ \end{pmatrix} \begin{bmatrix} -7.9 \\ 0.1 \\ -7.9 \\ -3.9 \\ 4.1 \\ -7.9 \\ -9.9 \end{bmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ \\ \\ \\ \\ \end{pmatrix}$$

The output obtained is fed back to the network to get the next fragment. This process continues until the entire memory is retrieved.

B-Matrix * Fragment Vector = Weighted Sum T_i = Output

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 3 & -1 & -1 & 0 & 0 & 0 & 0 \\ 1 & -3 & 1 & -1 & 0 & 0 & 0 \\ 1 & 1 & -3 & 3 & -3 & 0 & 0 \\ -1 & -1 & 3 & -3 & 3 & -5 & 0 \end{pmatrix} * \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} \\ 1 \\ \\ \\ \\ \\ \end{pmatrix} \begin{bmatrix} -7.9 \\ 0.1 \\ -7.9 \\ -3.9 \\ 4.1 \\ -7.9 \\ -9.9 \end{bmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ \\ \\ \end{pmatrix}$$

B-Matrix * Fragment Vector = Weighted Sum T_i = Output

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 3 & -1 & -1 & 0 & 0 & 0 & 0 \\ 1 & -3 & 1 & -1 & 0 & 0 & 0 \\ 1 & 1 & -3 & 3 & -3 & 0 & 0 \\ -1 & -1 & 3 & -3 & 3 & -5 & 0 \end{pmatrix} * \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} \\ -2 \\ \\ \\ \\ \\ \end{pmatrix} \begin{bmatrix} -7.9 \\ 0.1 \\ -7.9 \\ -3.9 \\ 4.1 \\ -7.9 \\ -9.9 \end{bmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ -1 \\ \\ \end{pmatrix}$$

B-Matrix * Fragment Vector = Weighted Sum T_i = Output

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 3 & -1 & -1 & 0 & 0 & 0 & 0 \\ 1 & -3 & 1 & -1 & 0 & 0 & 0 \\ 1 & 1 & -3 & 3 & -3 & 0 & 0 \\ -1 & -1 & 3 & -3 & 3 & -5 & 0 \end{pmatrix} * \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ -1 \end{pmatrix} = \begin{pmatrix} \\ \\ \\ \\ 5 \\ \\ \end{pmatrix} \begin{bmatrix} -7.9 \\ 0.1 \\ -7.9 \\ -3.9 \\ 4.1 \\ -7.9 \\ -9.9 \end{bmatrix} = \begin{pmatrix} -1 \\ 1 \\ 1 \\ -1 \\ -1 \\ 1 \\ \end{pmatrix}$$

B-Matrix * Fragment Vector | Weighted Sum T_i | Output

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 3 & -1 & -1 & 0 & 0 & 0 & 0 \\ 1 & -3 & 1 & -1 & 0 & 0 & 0 \\ 1 & 1 & -3 & 3 & -3 & 0 & 0 \\ -1 & -1 & 3 & -3 & 3 & -5 & 0 \end{pmatrix} * \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ -1 \\ 1 \end{pmatrix} = \begin{pmatrix} \\ \\ \\ \\ \\ \\ -10 \end{pmatrix} \begin{bmatrix} -7.9 \\ 0.1 \\ -7.9 \\ -3.9 \\ 4.1 \\ -7.9 \\ -9.9 \end{bmatrix} = \begin{pmatrix} -1 \\ 1 \\ 1 \\ -1 \\ -1 \\ 1 \\ -1 \end{pmatrix}$$

After repeated application of the fragment to the network the entire memory vector X^1 could be retrieved from the network. Similar procedure is followed for all the other memories that were stored into the network and results are displayed below.

B-Matrix * Fragment Vector | Weighted Sum T_i | Output

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 3 & -1 & -1 & 0 & 0 & 0 & 0 \\ 1 & -3 & 1 & -1 & 0 & 0 & 0 \\ 1 & 1 & -3 & 3 & -3 & 0 & 0 \\ -1 & -1 & 3 & -3 & 3 & -5 & 0 \end{pmatrix} * \begin{pmatrix} 1 \\ -1 \\ -1 \\ \\ \\ \\ \end{pmatrix} = \begin{pmatrix} \\ 5 \\ \\ \\ \\ \\ \end{pmatrix} \begin{bmatrix} -7.9 \\ 0.1 \\ -7.9 \\ -3.9 \\ 4.1 \\ -7.9 \\ -9.9 \end{bmatrix} = \begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \\ \\ \\ \end{pmatrix}$$

B-Matrix * Fragment Vector | Weighted Sum T_i | Output

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 3 & -1 & -1 & 0 & 0 & 0 & 0 \\ 1 & -3 & 1 & -1 & 0 & 0 & 0 \\ 1 & 1 & -3 & 3 & -3 & 0 & 0 \\ -1 & -1 & 3 & -3 & 3 & -5 & 0 \end{pmatrix} * \begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \\ \\ \\ \end{pmatrix} = \begin{pmatrix} \\ \\ 2 \\ \\ \\ \\ \end{pmatrix} \begin{bmatrix} -7.9 \\ 0.1 \\ -7.9 \\ -3.9 \\ 4.1 \\ -7.9 \\ -9.9 \end{bmatrix} = \begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \\ -1 \\ \\ \end{pmatrix}$$

B-Matrix * Fragment Vector = Weighted Sum T_i = Output

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 3 & -1 & -1 & 0 & 0 & 0 & 0 \\ 1 & -3 & 1 & -1 & 0 & 0 & 0 \\ 1 & 1 & -3 & 3 & -3 & 0 & 0 \\ -1 & -1 & 3 & -3 & 3 & -5 & 0 \end{pmatrix} * \begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \\ -1 \\ \\ \end{pmatrix} = \begin{pmatrix} \\ \\ \\ \\ \\ 9 \\ \end{pmatrix} \begin{bmatrix} -7.9 \\ 0.1 \\ -7.9 \\ -3.9 \\ 4.1 \\ -7.9 \\ -9.9 \end{bmatrix} = \begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \\ -1 \\ 1 \\ \\ \end{pmatrix}$$

B-Matrix * Fragment Vector = Weighted Sum T_i = Output

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 3 & -1 & -1 & 0 & 0 & 0 & 0 \\ 1 & -3 & 1 & -1 & 0 & 0 & 0 \\ 1 & 1 & -3 & 3 & -3 & 0 & 0 \\ -1 & -1 & 3 & -3 & 3 & -5 & 0 \end{pmatrix} * \begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \\ -1 \\ 1 \\ \\ \end{pmatrix} = \begin{pmatrix} \\ \\ \\ \\ \\ -14 \\ \end{pmatrix} \begin{bmatrix} -7.9 \\ 0.1 \\ -7.9 \\ -3.9 \\ 4.1 \\ -7.9 \\ -9.9 \end{bmatrix} = \begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \end{pmatrix}$$

With repeated application of a fragment vector, memory X^2 could be retrieved from the network.

Applying the same approach for retrieving X^3:

B-Matrix * Fragment Vector = Weighted Sum T_i = Output

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 3 & -1 & -1 & 0 & 0 & 0 & 0 \\ 1 & -3 & 1 & -1 & 0 & 0 & 0 \\ 1 & 1 & -3 & 3 & -3 & 0 & 0 \\ -1 & -1 & 3 & -3 & 3 & -5 & 0 \end{pmatrix} * \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \\ \\ \\ \end{pmatrix} = \begin{pmatrix} \\ \\ \\ \\ \\ 6 \\ \end{pmatrix} \begin{bmatrix} -7.9 \\ 0.1 \\ -7.9 \\ -3.9 \\ 4.1 \\ -7.9 \\ -9.9 \end{bmatrix} = \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ \\ \\ \end{pmatrix}$$

B-Matrix * Fragment Vector Weighted Sum T_i Output

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 3 & -1 & -1 & 0 & 0 & 0 & 0 \\ 1 & -3 & 1 & -1 & 0 & 0 & 0 \\ 1 & 1 & -3 & 3 & -3 & 0 & 0 \\ -1 & -1 & 3 & -3 & 3 & -5 & 0 \end{pmatrix} * \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ \\ \end{pmatrix} = \begin{pmatrix} \\ \\ \\ \\ \\ \\ -9 \end{pmatrix} \begin{bmatrix} -7.9 \\ 0.1 \\ -7.9 \\ -3.9 \\ 4.1 \\ -7.9 \\ -9.9 \end{bmatrix} = \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \\ \\ \end{pmatrix}$$

B-Matrix * Fragment Vector Weighted Sum T_i Output

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 3 & -1 & -1 & 0 & 0 & 0 & 0 \\ 1 & -3 & 1 & -1 & 0 & 0 & 0 \\ 1 & 1 & -3 & 3 & -3 & 0 & 0 \\ -1 & -1 & 3 & -3 & 3 & -5 & 0 \end{pmatrix} * \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \\ \\ \end{pmatrix} = \begin{pmatrix} \\ \\ \\ \\ \\ \\ 14 \end{pmatrix} \begin{bmatrix} -7.9 \\ 0.1 \\ -7.9 \\ -3.9 \\ 4.1 \\ -7.9 \\ -9.9 \end{bmatrix} = \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ \end{pmatrix}$$

From the above X^3 could be retrieved from the network. Applying the same procedure to retrieve X^4:

B-Matrix * Fragment Vector Weighted Sum T_i Output

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 3 & -1 & -1 & 0 & 0 & 0 & 0 \\ 1 & -3 & 1 & -1 & 0 & 0 & 0 \\ 1 & 1 & -3 & 3 & -3 & 0 & 0 \\ -1 & -1 & 3 & -3 & 3 & -5 & 0 \end{pmatrix} * \begin{pmatrix} -1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \\ \\ \end{pmatrix} = \begin{pmatrix} \\ \\ \\ \\ \\ \\ 8 \end{pmatrix} \begin{bmatrix} -7.9 \\ 0.1 \\ -7.9 \\ -3.9 \\ 4.1 \\ -7.9 \\ -9.9 \end{bmatrix} = \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ \end{pmatrix}$$

For the example considered all the memories stored in the network could be retrieved when variable thresholds are applied over the B-Matrix approach.

This experiment has been performed several times considering various inputs and the results are displayed below in a table.

Neurons	Memories	Memories Stored	Fixed Threshold	Variable Threshold
10	10	1	1	1
20	10	4	2	3
30	10	7	3	6
40	10	8	3	8
50	10	10	3	8
60	10	10	7	8
70	10	10	5	7
80	10	9	8	9
90	10	10	9	10
100	10	10	10	10

Table 2. Data for number of memories retrieved using fixed and variable threshold approach when 10 memories and neurons from 10-100 are considered.

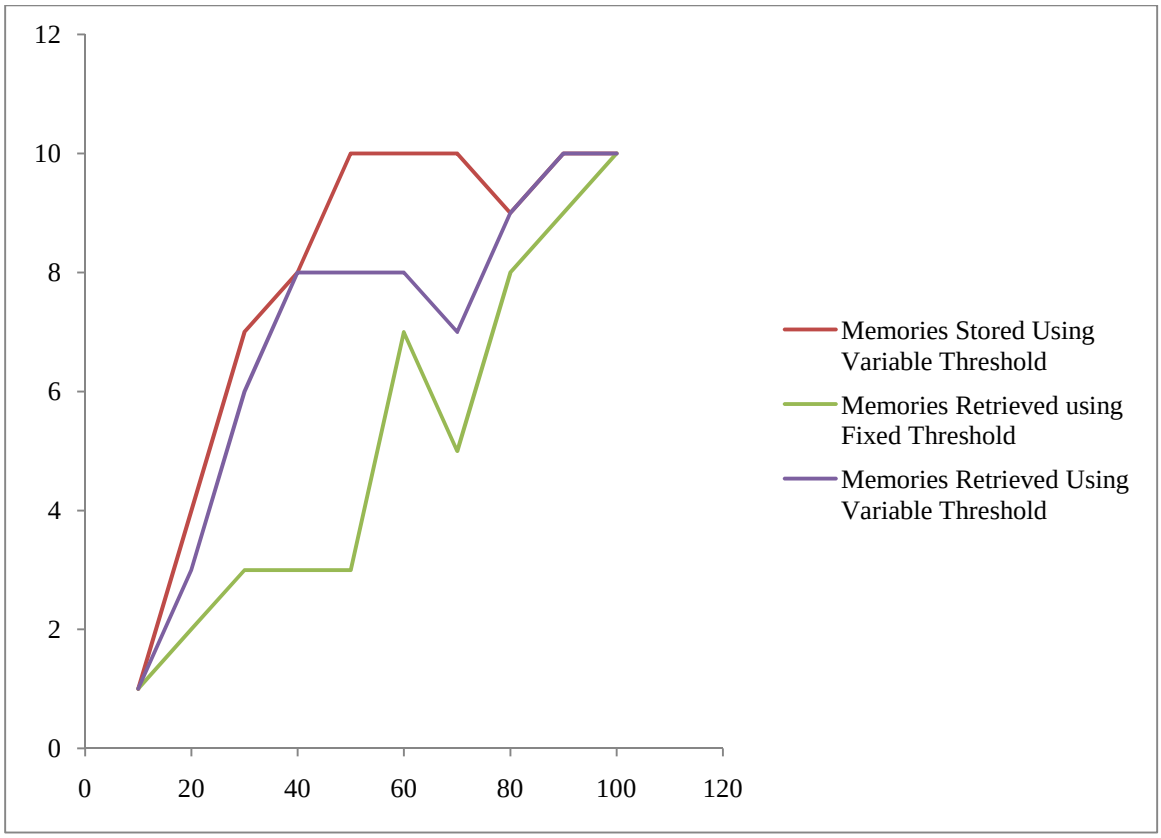

Figure 9. Graph representing the improvement in memory retrieval from a neural network when variable threshold is used.

From the graph above we can observe that when the variable threshold approach is applied to the B-Matrix approach of memory retrieval there is an increase in the percentage of memories retrieved from the network.

Quaternary/ Non-Binary Neural Network:

In all the topics discussed above we have used a simplified assumption of a neural network i.e. a binary neural network. It is not easy to compare these binary neural networks to the neural networks in biological organisms, since the biological neurons not only carry electrical impulses of varying magnitude, but they are also associated with a train of voltage spikes. For example most images are not binary and conversion of these images to binary would cause a loss of information. Non-binary or *n-ary* neural networks were proposed [14] motivated by the

complexity of biological neural networks. The next state of the neuron is determined by equations like

$$x_i = \sum_j T_{i,j} V_j$$

Consider the construction of a quaternary neural network [14] instead of a binary neural network. Such a network implements the same principles, as does a binary one, with the exception that the neurons now, map to a larger set of activations.

$$V_i = \begin{cases} -4 & x_i < -t \\ -1 & x_i < 0 \\ 1 & x_i < t \\ 4 & x_i > t \end{cases}$$

Here t, 0 and –t are the thresholds. To store maximum patterns in the network a learning algorithm similar to the delta rule can be applied to the network to modify the synaptic weights.

$$\Delta T_{ij} = c\,(V^s_i - V_i)\,V^s_j$$

where V_i is the output of neuron I when pattern V^s is applied to the network and c is the learning constant. if $V^s_i = Vi$ then none of the weights in the i^{th} row need to be adjusted. Repeated application of this procedure will lead to a convergence of V_i to V^s_i provided that the convergence ratio t/c is large enough. If convergence ratio is large, the equation may have to be applied many number of times to learn the pattern and if the convergence ratio is too small than it may not be even possible to learn the pattern.

Storage capacity of a Neural Network:

The storage capacity of a binary neural network according to the Hopfield network is 0.14N where N is the number of neurons considered. The storage capacity of a non-binary neural

network is much higher than that of a binary neural network and so is the complexity associated in dealing with a non-binary neural network.

In a quaternary binary neural network the selection of the threshold value plays a vital role in determining the storage capacity. For any n-ary neural network determination of convergence ratio t/c is given by

$$t/c=V_{max}^{2}\,Vdiff(N-1)$$

where V_{max} is the maximum allowable magnitude for the output of a neuron and Vdiff is maximum difference between any two "adjacent" output values.

For the quaternary model this formula gives the minimum value of t/c that will guarantee that any pattern can be stored.

Calculating the threshold ‘t’ with the above for network with inputs 2, -2, 1, -1 would be 48. Experiments have been performed on 7 neurons. Each time a different pattern is considered. The value of learning constant c is kept 1 for all the experiments performed. The value of t has been varied over a large range. In the first test performed the number of neurons was set to 7. For each value of t, 100 attempts have been made to store 1 to six random patterns. Similar test was also performed taking 9 neurons into consideration. The output values in both the cases are considered to be -4, -1, 1, and 4. The results of the test are displayed in the table below.

t/c	Patterns					
	1	2	3	4	5	6
96	97	93	93	88	82	57
144	100	94	97	93	69	46
192	100	97	97	87	69	31
240	100	100	96	83	53	25
288	100	100	89	77	54	13
336	100	98	81	58	36	11

Table 3. Results of non-binary network capacity with 7 neurons

t/c	Patterns					
	1	2	3	4	5	6
192	88	45	21	11	2	0
240	93	86	82	77	75	57
288	97	93	83	78	68	33
336	91	92	83	81	53	7
384	100	98	89	63	49	11
432	100	92	87	69	29	9

Table 4. Results of non-binary network capacity with 9 neurons

In the first experiment about 53% we were successful in storing patterns and were around 50% successful when nine neurons were considered. From the above we can observe that for a given number of neurons the quaternary network could store more neurons than the binary model.

Variable Threshold Approach for non-binary Neural Network:

Applying the variable threshold approach in storing memories in a non-binary neural network requires the identification of threshold at each neuron in an iterative process such that maximum numbers of memories are stored in the network. In the case of non-binary neural network the calculation of threshold at each neuron gets very complex as the number of iterations required to determine the threshold at each neuron increases. Consider a simple example of a network with five neurons. The T-Matrix is given by

T Matrix:

$$\begin{pmatrix} 0 & -4 & 16 & -4 & -4 \\ -4 & 0 & -4 & 1 & 1 \\ 16 & -4 & 0 & -4 & -4 \\ -4 & 1 & -4 & 0 & 1 \\ -4 & 1 & -4 & 1 & 0 \end{pmatrix}$$

To store a memory vector given by

X1= -4 1 -4 1 1

Initially the threshold values at each neuron are to be calculated using the same procedure used for a binary neural network. For the network considered above the threshold values are found to be:

-75.9 Threshold at Neuron 1

34.1 Threshold at Neuron 2

-75.9 Threshold at Neuron 3

34.1 Threshold at Neuron 4

34.1 Threshold at neuron 5

With these threshold values the memory vector considered above could be stored into the network.

$$\begin{pmatrix} 0 & -4 & 16 & -4 & -4 \\ -4 & 0 & -4 & 1 & 1 \\ 16 & -4 & 0 & -4 & -4 \\ -4 & 1 & -4 & 0 & 1 \\ -4 & 1 & -4 & 1 & 0 \end{pmatrix} * \begin{pmatrix} -4 \\ 1 \\ -4 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} -76 \\ 34 \\ -76 \\ 34 \\ 34 \end{pmatrix} \begin{bmatrix} -75.9 \\ 34.1 \\ -75.9 \\ 34.1 \\ 34.1 \end{bmatrix} = \begin{pmatrix} -4 \\ 1 \\ -4 \\ 1 \\ 1 \end{pmatrix}$$

Similar experiments have been performed to assess the storage capacity of a non-binary neural network when variable threshold approach is used. 100 patterns have been attempted to store in the network with 1 and 2 patterns considered at a time. This experiment has been conducted by taking a network of five neurons.

Number of neurons	1 Pattern	2 Patterns	3 Patterns
5	96	62	29
6	91	51	17

Table 5. Number of patterns stored in the network when variable threshold approach is used.

The computational complexity in case of variable threshold approach for the non-binary neural network increases because the number of iterations in determining the threshold at each neuron is much higher. The number of iterations greatly depend on the input vectors considered and the values for the output considered i.e. (-4,-1,1,4) or (-3,-1,1,3) etc., therefore the selection of an appropriate threshold increase rate for each iteration plays a very important role.

CHAPTER IV

EXPERIMENTAL RESULTS

Introduction:

The use of variable threshold approach has been simulated in Java. The simulation takes in the number of neurons and memories to be considered and generates random numbers for the memory vector. From these randomly generated memory vectors we construct the T-Matrix.

Initially the simulation calculates the number of memories stored into the network using the fixed threshold approach. After finding the number of memories stored using the fixed threshold approach, the variable threshold approach is applied to the same set of memory vectors considered previously for the fixed threshold approach.

In finding the number of memories stored in the network using the variable threshold approach initially the threshold corresponding to each neuron in the network is calculated. Once the thresholds at each neuron is determined by a iterative approach the memories are stored in the network if

$$x = \mathrm{Ti}(W.x)$$

where T_i is the threshold at the i^{th} neuron.

Experiment 1: Results for Variable Threshold Neural Network:

Neurons	Memories	Fixed Threshold	Variable Threshold
10	10	1	1
20	10	2	3
30	10	3	7
40	10	3	8
50	10	3	10
60	10	7	10
70	10	5	10
80	10	8	9
90	10	10	10
100	10	10	10

Table 6. Data for number of memories stored using fixed and variable threshold.

In the table above 10 memories are stored into the network with neurons from 10-100 are considered during each experiment. Each time the experiment is performed, different set of memories are considered.

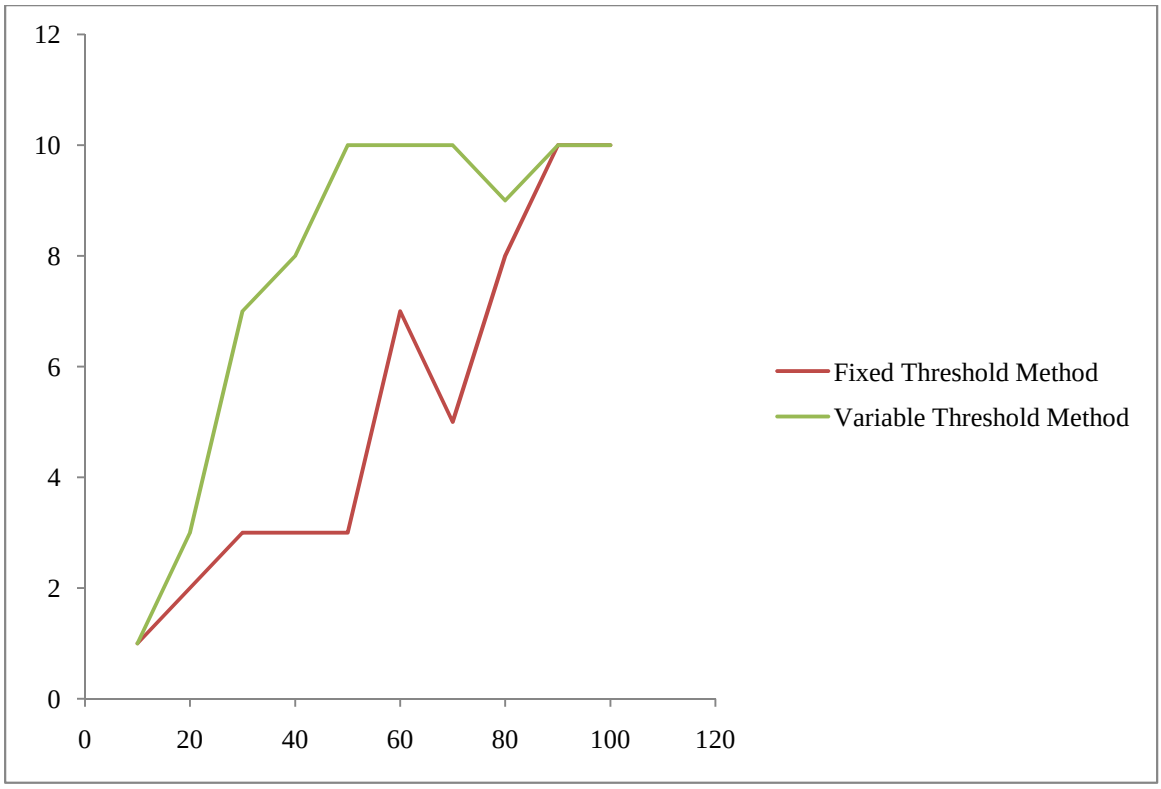

Figure 10. Fixed Threshold vs. variable threshold memory storage in a neural network when 10 memories are considered.

From the graph above we can clearly observe that when neurons from 10- 100 are considered in storing ten memories, the use of a variable threshold approach could store more memories into the network.

Experiment 2: Results for Variable Threshold Neural Network:

Neurons	Memories	Fixed Threshold Method	Variable Threshold Method
20	20	2	2
40	20	0	3
60	20	2	8
80	20	3	12
100	20	4	12
120	20	10	19
140	20	17	20
160	20	13	19
180	20	17	20
200	20	16	20

Table 7. Data for number of memories stored using fixed and variable threshold when 20 memories are considered.

In the table above 20 memories are stored into the network with neurons from 20-200 are considered during each experiment. Each time the experiment is performed, different set of memories are considered.

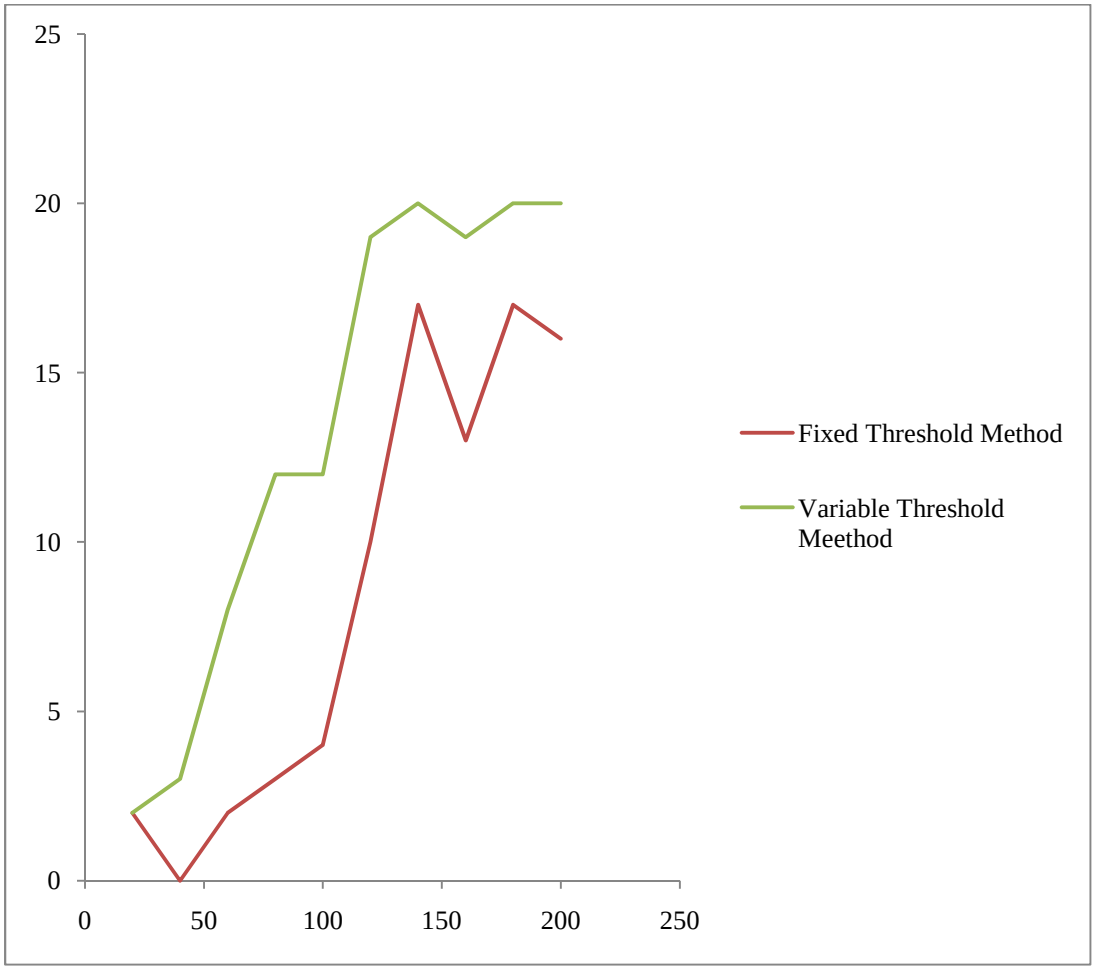

Figure 11. Fixed threshold vs. variable threshold memory storage in a neural network when 20 memories are considered.

From the graph above we can clearly observe that when neurons from 20- 200 are considered in storing twenty memories, the use of a variable threshold approach could store more memories into the network.

Experiment 3: Results for Variable Threshold Neural Network:

Neuron	Memory	Fixed Threshold	Variable Threshold
100	50	0	0
150	50	0	1
200	50	1	5
250	50	5	15
300	50	10	31
350	50	13	41
400	50	27	46
450	50	28	49
500	50	29	49

Table 8. Data for number of memories stored using fixed and variable threshold when 50 memories are considered.

In the table above 50 memories are stored into the network with neurons from 50-500 are considered during each experiment. Each time the experiment is performed, different set of memories are considered.

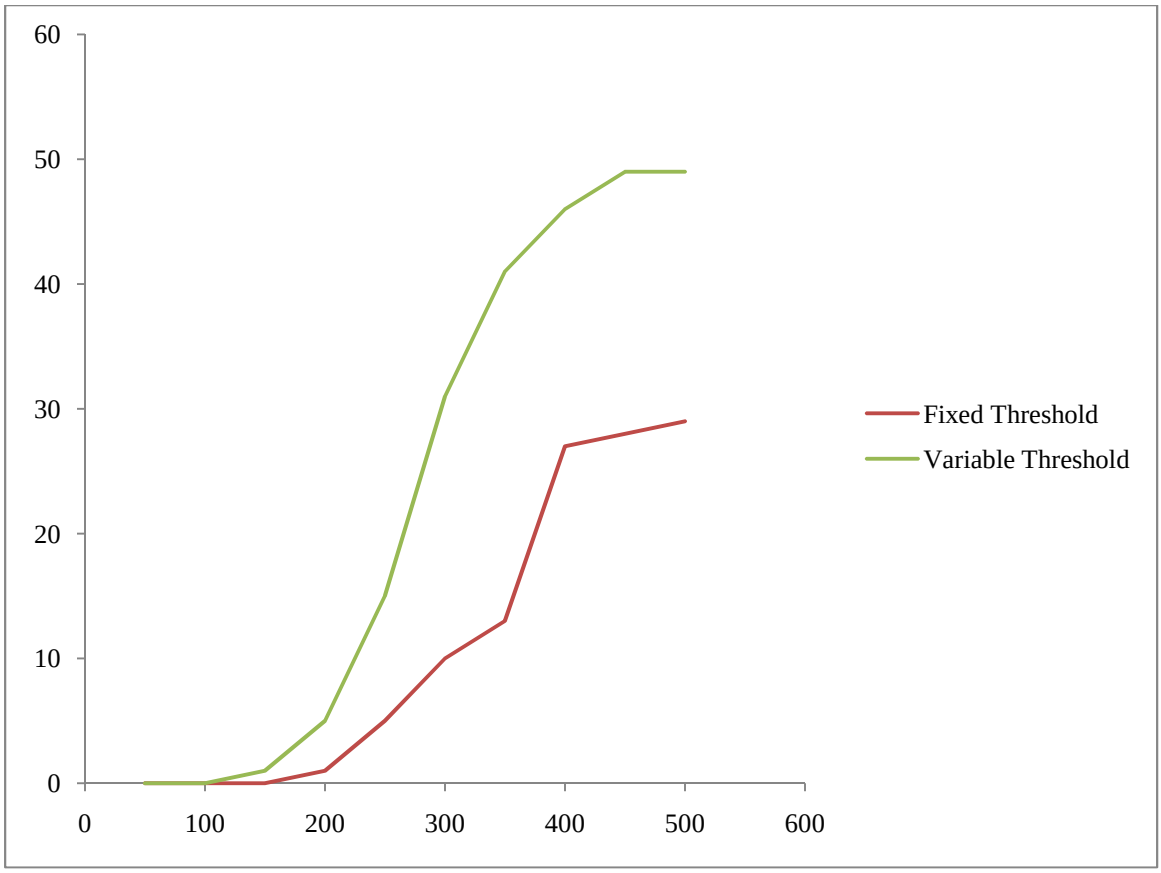

Figure 12. Fixed threshold vs. variable threshold memory storage in a neural network when 50 memories are considered.

From the graph above we can clearly observe that when neurons from 50- 500 are considered in storing fifty memories, the use of a variable threshold approach could store more memories into the network.

Experiment 4: Results for Variable Threshold Neural Network:

Neuron	Memory	Fixed Threshold	Variable Threshold
400	100	0	0
450	100	0	1
500	100	1	8
550	100	3	18
600	100	2	30
650	100	3	31
700	100	16	52
750	100	9	58
800	100	25	68
850	100	27	78
900	100	47	88
925	100	43	93
975	100	43	94
1000	100	61	95

Table 9. Data for number of memories stored using fixed and variable threshold when 100 memories are considered.

From the graph above we can clearly observe that when neurons from 100- 1000 are considered in storing hundred memories, the use of a variable threshold approach could store more memories into the network.

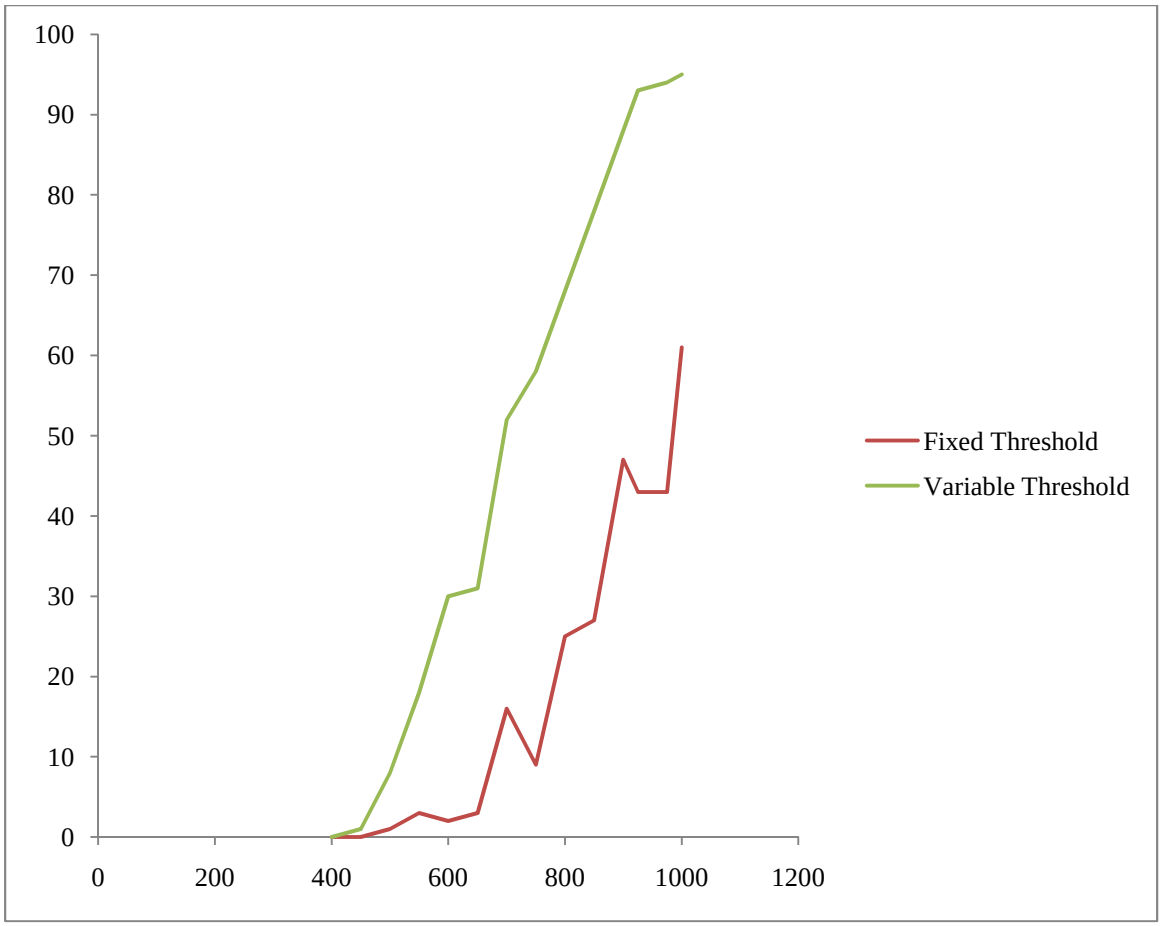

Figure 13. Fixed threshold vs. variable threshold memory storage in a neural network when 100 memories are considered.

From all the experiments performed with 100 neurons to 1000 neurons we can observe that, from smaller networks to larger networks, when variable threshold algorithm results in storing a larger percentage of memories into the network when compared to fixed threshold approach.

CHAPTER V

CONCLUSION

In this thesis we describe the use of the variable threshold learning approach, which helps in increasing the storage capacity of a neural network. The issues involved in determining the thresholds at each neuron in the network have been described. The variable threshold approach has been applied on a neural network with hundred memories and thousand neurons and the increase in percentage of memory storage has been noted to be approximately around 15%. This method has been further extended to the B-Matrix approach of memory retrieval. Future work involves further experiments on the behavior of this approach on larger networks.

REFERENCES

1 D.O. Hebb, *The Organization of Behavior*. Wiley, 1949.

2 S. Kak, Active agents, intelligence, and quantum computing. *Information Sciences*, vol. 128, pp. 1-17, 2000.

3 S. Kak, The three languages of the brain: quantum, reorganizational, and associative. In: K. Pribram and J. King, Editors, *Learning as Self-Organization*, Lawrence Erlbaum, Mahwah (1996), pp. 185–219.

4 J.J. Hopfield, Neural networks and physical systems with emergent collective computational properties. *Proc. Nat. Acad. Sci. (USA)*, vol. 79, pp. 2554-2558, 1982.

5 S. Kak, Feedback neural networks: new characteristics and a generalization. *Circuits, Systems, and Signal Processing*, vol. 12, pp. 263-278, 1993.

6 S. Kak and M.C. Stinson, A bicameral neural network where information can be indexed. *Electronics Letters*, vol. 25, pp. 203-205, 1989.

7 S. Kak, On training feedforward neural networks. Pramana, vol. 40, pp. 35-42, 1993.

8 S. Kak, New training algorithm in feedforward neural networks, First International Conference on Fuzzy Theory and Technology, Durham, N. C., October 1992. Also in Wang, P.P. (Editor), Advance in fuzzy theory and technologies, Durham, N. C. Bookwright Press, 1993.

9 S. Kak, New algorithms for training feedforward neural networks, Pattern Recognition Letters, vol. 15, pp. 295-298, 1994.

10 S. Kak, A class of instantaneously trained neural networks, Information Sciences, 148, 97-102, 2002.

11 M.C. Stinson and S. Kak, Bicameral neural computing. *Lecture Notes in Computing and Control*, vol. 130, pp. 85-96, 1989.

12 Raul Rojas, Neural Networks : A Systematic Introduction. Springer, 1996.

13 D.L. Prados and S. Kak, Neural network capacity using the delta rule. *Electronics Letters*, vol. 25, pp. 197-199, 1989.

14 D. Prados and S. Kak, Non-binary neural networks. *Lecture Notes in Computing and Control*, vol. 130, pp. 97-104, 1989.

15 Colin Fyfe, Hebbian Learning and Negative Feedback Networks, Springer, 2005.

16 S Kak, Single Neuron Memories and the Network's Proximity Matrix. 2009. arXiv:0906.0798

17 K.C.Lingashetty, Active Sites model for the B-Matrix Approach, 2010. arXiv:1006.4754v1

18 S. Kak, Multilayered array computing. Information Sciences 45, 347-365, 1988.

19 S. Kak, A two-layered mesh array for matrix multiplication. Parallel Computing 6, 383-385, 1988.

20 S. Kak, On the mesh array for matrix multiplication. 2010. arXiv:1010.

VITA

Praveen Kuruvada

Candidate for the Degree of

Master of Science

Thesis: MEMORY STORAGE IN A VARIABLE THRESHOLD ARTIFICIAL NEURAL NETWORK

Major Field: Computer Science

Biographical:

Education:

Completed the requirements for the Master of Science in your Computer Science at Oklahoma State University, Stillwater, Oklahoma in July, 2011.

Completed the requirements for the Bachelor of Technology in your Information Technology at Jawaharlal Nehru Technological University, Hyderabad, India in 2009.

Experience:

Graduate Assistant: Plant & Soil Science, OSU Dec 2009- May 2011

- Managed network drive and access restrictions to these drives. Updated and maintained the server software.
- Responsible for solving software related issues and providing new system recommendations.

Name: Praveen Kuruvada Date of Degree: July, 2011

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study: MEMORY STORAGE IN A VARIABLE THRESHOLD ARTIFICIAL NEURAL NETWORK

Pages in Study: 045 Candidate for the Degree of Master of Science

Major Field: Computer Science

Scope and Method of Study:

This thesis is concerned with the problem of memory storage in a feedback neural network. A new learning approach has been developed that assigns variable thresholds to the neurons through an iterative process. This approach has been further applied to B-Matrix approach of memory retrieval. Experiments have also been conducted on non-binary neural networks and their storage capacity has been evaluated.

Findings and Conclusions:

Through simulations and analysis, it has been demonstrated that use of the variable threshold learning increases the storage capacity of a network by about 15%. A similar increase is found in the memory retrieval process by the use of variable thresholds for B-Matrix approach.

ADVISER'S APPROVAL: Dr. Subhash Kak