ACCELERATED CONVERGENCE IN BACK PROPAGATION LEARNING ALGORITHM

A THESIS

SUBMITTED TO THE DEPARTMENT OF

ELECTRICAL AND ELECTRONIC ENGINEERING, BUET

IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF

MASTER OF SCIENCE IN ENGINEERING

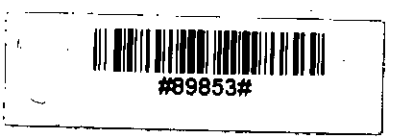
BY

ARUN KUMAR SAHA

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY

JANUARY, 1996



The thesis titled, "Accelerated Convergence in Back-propagation Learning Algorithm" submitted by Arun Kumar Saha, Roll No.:901307F, Registration No.: 84142 of M.Sc. in Engineering has been accepted as satisfactory in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering.

1. S. Kamruzzaman 15.1.96

Dr. Joarder Kamruzzaman — Chairman

Assistant Professor — (Supervisor)

Depertment of Electrical and Electronic Engg.

BUET, Dhaka-1000.

2. ABM Siddique Hossain 15/1/96

Dr. A.B.M. Siddique Hossain — Member

Professor And Head — (Ex-Officio)

Depertment of Electrical and Electronic Engg.

BUET, Dhaka-1000.

3. 15/1/96

Dr. Mashiur Rahman Bhuiyan — Member

Assistant Professor

Depertment of Electrical and Electronic Engg.

BUET, Dhaka-1000.

4. 15/1/96

Dr. M. Kaykobad — Member

Associate Professor — (External)

Depertment of Computer Science and Engg.

BUET, Dhaka-1000.

DEDICATED

TO

PARENTS

Declaration

I hereby declare that this thesis work has not been submitted elsewhere for the award of any degree or diploma or for publication.

Countersigned:

J. Kamruzzaman

Dr. Joarder Kamruzzaman

Arun Kumar Saha

Arun Kumar Saha

CONTENTS

Acknowledgment

The Author expresses his sincere and profound gratitude to Dr. Joarder Kamruzzaman, Assistant Professor of the Department of Electrical and electronic Engineering, Bangladesh University of Engineering technology (BUET), Dhaka for his incessant and meticulous guidance in completing this work. The author thanks him for his outstanding suggestions and help during vital phases of the work. He with his thoughtfulness and patience taught the author how to carry out a research work and thus have prepared him for future endeavors.

The author wishes to express his thanks and regards to Dr. A. B. M. Siddique Hossain, Professor and Head of the Department of Electrical and Electronic Engineering, BUET for his support and co-operation he received from him and the department.

The author also wishes to express his thanks to Dr. Mohammad Ali Chowdhury and Md. Nasim Ahmed Dewan for all the assistance and support they provided to carry out this research work. Sincerest thanks to Mr. Ajit Chandra Dutta, Executive Engineer, G.M.D, BPDB, Srimangal, Moulvibazar, for his helpful suggestions, co-operation and encouragement.

Abstract

Recent developments in the field of artificial neural networks have provided potential applications in various fields. Artificial neural nets are inspired from the studies of biological nervous system and composed of many simple nonlinear computational elements called neurons which are connected by links of variable weights. These weights are adjusted by a learning process and finally settle down to a set of weights that realizes the task at hand. The most popular learning algorithm for feed-forward connectionist networks is Back-propagation algorithm This algorithm defines sum of squared errors measured at the output layer as error function and updates weights to minimize this error by steepest descent method. Major drawbacks of this algorithm are its slow convergence and possibility of getting stuck in local minima. These drawbacks hinder its widespread applications in real-world problems. Several methods have been proposed to improve its convergence. But these methods require some additional parameters to be adjusted and fast increase of some learning rate parameter might cause unstable behavior in the learning process and needs more complicated training procedure. Having considered all these problems, a new error function expressed as exponential of the sum of squared errors measured at the output layer is defined in the proposed research. Weight update using this modification varies the learning rate parameter dynamically during training as opposed to constant learning rate parameter used in standard Back-propagation. This adaptation of learning rate during learning is found to significantly improve the convergence speed of Back-propagation algorithm.

List of Figures

List of Tables

5.8 Comparison of training cycles required to realize sine function approximation problem with a 1-6-1 network using standard Back-propagation and proposed modification.

5.9 Comparison of training cycles required to realize sine function approximation problem with a 1-7-1 network using standard Back-propagation and proposed modification.

CHAPTER 1

INTRODUCTION

GENERAL INTRODUCTION

BIOLOGICAL NEURONS

ARTIFICIAL NEURAL NETWORKS

POTENTIAL APPLICATIONS OF NEURAL NETWORKS

SCOPE OF THE PRESENT WORK

THESIS ORGANIZATION

1.1 General Introduction

Over the past few decades computer designers, engineers, and programmers are making great efforts in devising an intelligent machine that can achieve human like performance in speech and visual pattern recognition, natural language processing, dealing with fuzzy situations etc. Obviously, the best and only one natural model is the human brain - the essence of all knowledge. One reasonable conjecture in this regard is that thinking about computation in terms brain metaphor rather than the digital computer metaphor will lead to insights into the nature of intelligent behavior.

Present day digital computers are capable of doing amazing feats. They can effortlessly store vast quantities of information. Their circuits operate in nanoseconds and can perform extensive arithmetic calculations without error. Human brain cannot approach these capabilities. On the other hand, humans routinely perform simple tasks such as walking, talking and commonsense reasoning. Biological neurons which operate in millisecond range are extremely slow devices when compared to their counterpart in digital computers. Yet, humans can perform extremely complex tasks, like interpreting a visual scene or understanding a sentence, in just one tenth of a second. In other words, we do in about a hundred steps what current computers cannot do in ten million steps. The underlying reason is that, unlike conventional computers, the brain contains a huge number of neurons - the information processing elements of biological nervous system, acting in parallel. This suggests that in search for solutions to intelligent problems, we need machines built on large number of processing elements computing in parallel.

Neurons are also failure prone biological devices. They are constantly dying, and their firing patterns are irregular. Components of digital computers, in contrast, must operate perfectly and failure of one component causes loss of information. Thus, if artificial networks can be built with essential properties of biological neurons, it would not be sensitive to the failure of a few components, perhaps by using redundancy and distributing information across a wide range of components. This network would be much more fault tolerant than a digital computer. Another thing humans seem to be able to do better than computers is dealing with fuzzy situations. We have very large memories of visual, auditory and problem solving episodes and one key operation in solving new problems in finding closest matches to old situations. Inexact matching is something brain style model seem to be good at because of the diffuse and fluid way in which knowledge is represented.

Digital computers perform real time processing in an "interrupt-driven" mode, i.e., events must set an "interrupt flag" to request the processor's attention. A serial processor will respond to an event according to prioritization of each request with respect to others which is a complex and time consuming process involving selection, activation and suspension of various programs at different times and there is always a time delay between the occurrence of an event and its response. But, if an artificial network is modelled after biological neurons, it will be able to act in "real time" by responding to the events as they occur.

All these features found in brain serve as a strong motivation for the idea of building an intelligent machine modelled after biological neurons, known as ***artificial neural networks***. However, artificial neural network models are not meant to duplicate but to abstract a few properties from what are known of brain's functioning based on the present understanding of biological nervous system. Understanding of brain's functioning is yet far

from complete and contains many points of controversy. Much of this uncertainty concerns the details of biological mechanisms rather than neural functions. The organization and information processing characteristics of several nerve nets are fairly well understood. Using those few properties that are known, the artificial networks, much simplified models of biological neurons are very effective in doing large amount of computation with a very little hardware.

1.2 *Biological Neurons*

Figure 1.1 shows a schematic diagram of two biological neurons in synaptic contact. Neurons are electrically active cells; they interact with one another through the flow of local electrical (ionic) currents between them. These ionic currents are driven by a voltage difference across the neuron's cell membrane. This voltage is normally maintained at a level called the *membrane resting potential* by a set of active biochemical pumps located in the membrane [1]. The resting potential for many types of neurons is about 60 millivolts (measured at the cell's interior with respect to its exterior). These pumps create chemical concentration gradients in three primary ionic species: potassium (k^+), sodium (Na^+) and chloride ion (Cl^-).

A neuron impulse consists of a rapid voltage change which occurs in a localized section of a neuron's membrane. This voltage change is produced by a flow of local ionic currents through the membrane. Once initiated, these flows propagate to the adjacent neurons along the length of nerve fibre. Once a signal is propagated, another impulse cannot be produced for some period of time. During this so called *refractory period*, the pumps in the membrane work to reestablish the membrane's resting potential [2]. Because

of this activity, neurons cannot produce more than about one thousand impulses per second. An impulse thus lasts a few milliseconds.

The main elements of the biological neural systems are:

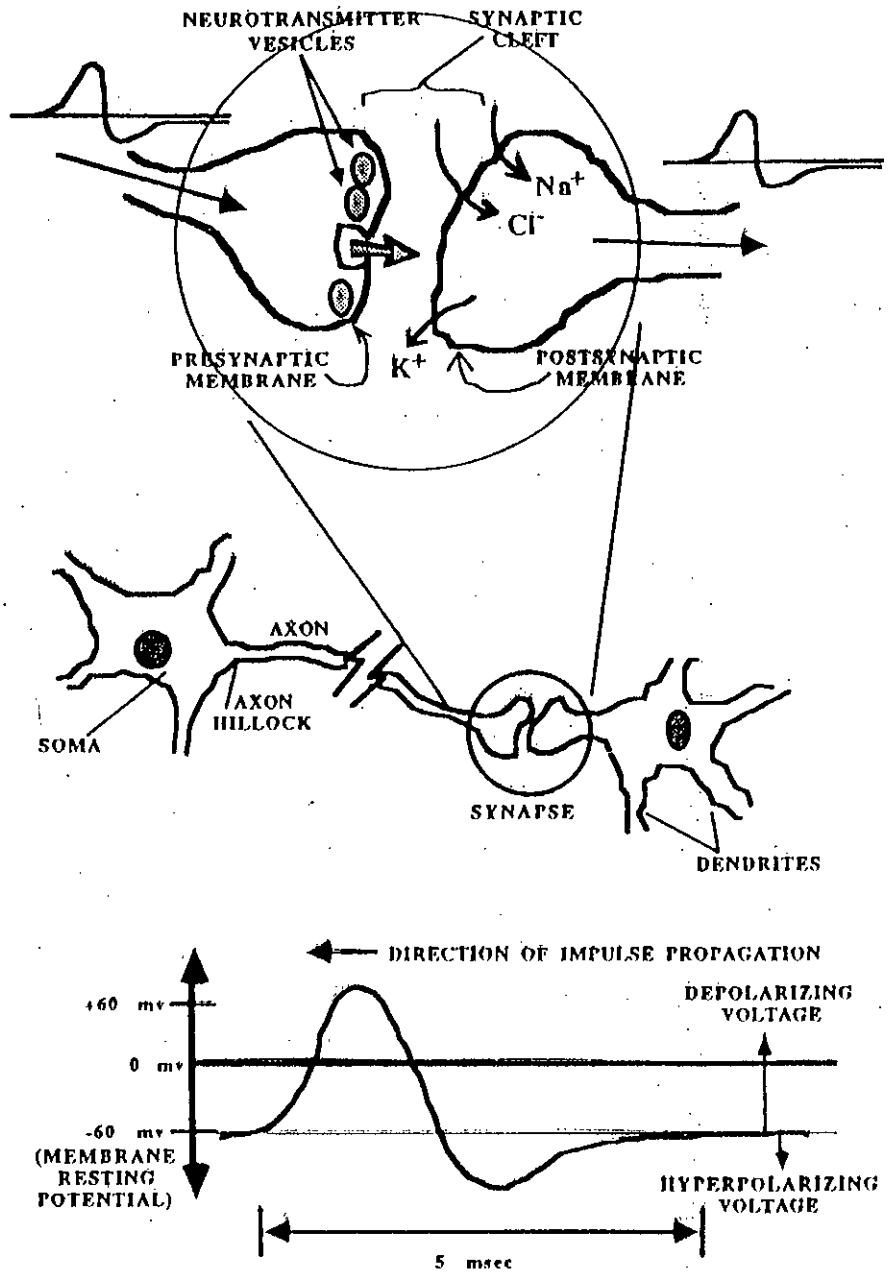


Figure 1.1 Two biological neurons in synaptic contact.

1. The *soma* or nerve cell, the large round central body of the neuron from 5 to 1000 microns in diameter.
2. The output fibre called the *axon*, attached to the soma, electrically active, produces the pulses which are emitted by the neuron.
3. The input fibre leading to the soma called *dendrite*, receives inputs from other neurons by means of specialized contacts.

Each such fibre transmits pulses of current (nerve impulses) in one direction, so that neurons feed currents to one another in definite directions. The axon of each cell feeds the dendrites of its sink-cells through *synapses*. Each such synapse consists of a pre-synaptic axon-end butted against a post-synaptic dendrite-end, usually with a small intervening gap called the *synaptic cleft*. As a nerve impulse reaches at axon-end, it causes release of a chemical called *neurotransmitter* which diffuses across the fluid filled synaptic gap. When this chemical reaches the dendrite end in a few msec, it induces a voltage change across the post-synaptic membrane of the dendrite. There are many types of neurotransmitter but a given transmitter always has two basic effects on the postsynaptic membrane. It can cause the membrane potential to become more positive which is termed as depolarizing the membrane or more negative which is termed as hyperpolarizing the membrane.

Neurons can combine impulses that reach them from other neurons in two integration effects called temporal summation and spatial summation [3]. *Temporal summation* is a mechanism that depends upon the pulse frequency of the incoming pulse train. A neuron's cell membrane acts like a leaky capacitor. Each incoming impulse tends to change the cell's membrane potential by some amount which gradually decays back toward zero. If an

impulse arrives so soon after a preceding one that the latter's voltage change has not completely decayed, then the two voltage changes add up. As a result, an input train of sufficiently high frequency will cause a continual increase or decrease in the recipient cell's membrane voltage and may become sufficiently depolarized to fire. Lower frequency input pulses may never manage to exceed the cell firing threshold. *Spatial summation* is an interaction that involves multiple inputs to a neuron. A neuron may receive thousands of inputs from other neurons. Some of these inputs may be so powerful that impulses arriving from single inputs may be sufficient to cause the recipient cell to fire. Spatial summation refers to roughly simultaneous arrival of pulses from a number of distinct inputs.

These simple mechanisms are surprisingly powerful. First, each neuron is basically an independent processor. Each neuron receives, combines and produces impulses as an autonomous information processing unit, operating in parallel with other neurons. The processing that occurs at each neural element is comparatively simple; but at any given instant, a tremendous number of such operations are in progress.

1.3 *Artificial Neural Networks*

Artificial neural networks are modelled after biological neurons to have the same kinds of information processing operations. A simple artificial neural system, an analogous to biological counterpart, is illustrated in Figure 1.2. Here neuron becomes independent processing element or node, the axons and dendrites become wires, and the synapses become variable resistors (links) carrying the weighted inputs that represent data or the sum of

weighted inputs from other processing elements.

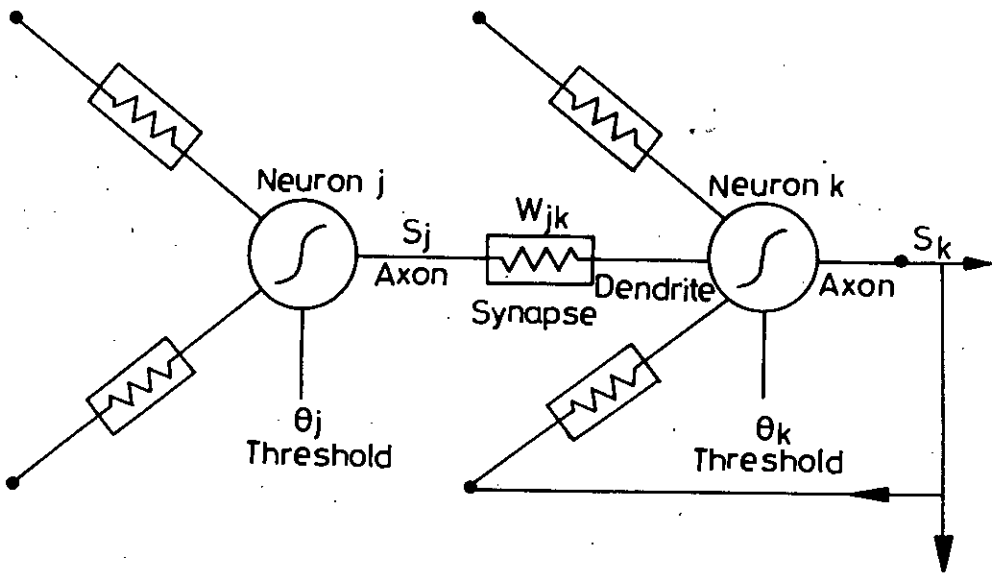


Figure 1.2 Two artificial neurons.

Artificial neural network is thus a parallel, distributed information processing structure consisting of processing elements interconnected via unidirectional signal channels called connection weights. These simple neuron-like processing elements receive, combine and produce outputs as autonomous information processing units and the connection weights between them encode the knowledge of the network after it is trained by some suitable learning algorithm designed to solve a specific problem.

1.4 *Potential Applications of Neural Networks*

Neural networks have far reaching potential as building blocks in tomorrow's intelligent computation. Already useful applications have been designed, built and commercialized, and much research continues in hopes of extending this success. Neural network applications emphasize areas where they appear to offer a more straightforward and easier approach than traditional computing and better performance as well. Neural networks offer possibilities for solving problems that require pattern recognition, pattern mapping, dealing with noisy data, pattern completion, associative look ups, and systems that learn or adapt during use. Examples of specific areas where these types of problems appear include speech synthesis and recognition, image processing and analysis, handwritten character recognition, sonar and seismic signal classification, associative memory, adaptive control etc. Some optimization tasks have also been addressed with neural networks.

The highly developed applications are hand written character recognition, text-to-speech converter (e.g, NETtalk), noise filtering in ECG, neural network based medical diagnosis system to assist doctors. A neural network based diagnosis system 'DESKNET' is able to diagnose 10 different skin diseases based on a set of 18 symptoms and test results. British Telecom, American Telegraph and Telecom (AT&T) and Nippon Telegraph and Telecom (NTT) are designing applications using neural network based speech processing, recognition and synthesis. Efforts are being made in designing voice-activated control using neural networks.

Identification of targets and industrial inspection provide a large domain for neural network applications. Machine fault diagnosis and analysis is an important commercial application of neural networks. Successful application studies have been done for jet, rocket and automotive engine

diagnosis. Neural networks are applied to a variety of financial analysis problems- such as credit assessment, financial forecasting and mortgage loan evaluations by companies that finance and insure mortgages. They are trained to forecast weather, clean up noise in video images, play games, forecast electric loads, identify signature and translate natural languages. Neural network along with fuzzy theory and genetic algorithm may play a new role in achieving the goal of building an intelligent machine in future.

1.5 *Scope of the Present Work*

The neural network approach to solve a problem requires definition of problem in terms of input-output relationship and a suitable learning algorithm to realize that relationship. Much of the efforts in neural network research has been devoted to developing algorithms which perform better and converges faster than the existing ones, and easier to implement in VLSI technology. One of the most popular algorithms is Back-propagation learning used to train multi-layer feedforward network. This algorithm is an iterative one that performs a steepest descent technique to reduce a sum-squared error measured at output layer. One of the limitations of Back-propagation algorithm is its slow convergence which hinders its widespread use in real time applications. The present work aims to accelerate the convergence of Back-propagation learning by defining a new cost function. This cost function is defined to be the exponential function of the sum-squared error of standard Back-propagation learning. With this cost function, the learning rate parameter is varied during training as opposed to constant learning rate parameter in standard Back-propagation learning. This adaptation of learning rate parameter is expected to speed up the learning process. Simulations using different problems will be done. Convergence

characteristics with this modification will be investigated and comparisons with standard Back-propagation will be made.

1.6 *Thesis Organization*

In chapter 2, formulation of standard Back-propagation algorithm and limitations of this algorithm are presented. In chapter 3, some notable previous works on accelerating the convergence of Back-propagation are illustrated. In chapter 4, the proposed acceleration technique is formulated. In chapter 5, simulation results are given. Conclusions and possible further investigations on the present work are mentioned in chapter 6.

CHAPTER 2

BACK-PROPAGATION LEARNING ALGORITHM

INTRODUCTION

FORMULATION OF BACK-PROPAGATION ALGORITHM

LIMITATIONS OF BACK-PROPAGATION ALGORITHM

REASONS FOR SLOW CONVERGENCE

SUMMARY

2.1 *Introduction*

Back-propagation is one of the most widely-used algorithms to train multilayer feed forward networks. Its weight update procedure is intuitively appealing because it is based on a relatively simple concept. If the network gives the wrong answer, then the weights are corrected so that the error is lessened and as a result future responses of the network are more likely to be correct.

Back-propagation networks are usually multiple layered, with each layer fully connected to the layers below and above. When the network is given an input, the updating of activation values propagates forward from the input layer of the processing units, through each internal layer, often called hidden layer, to the output layer of processing units. The output units then provide the network's response. When the network corrects its internal parameters, the correction mechanism starts with the output units and back-propagates backward through each hidden layer to the input layer - hence the term back-error propagation, or back-propagation.

Back-propagation is a tremendous step forward compared to its predecessor, the perceptron. The perceptron was limited to only two layers of processing units, with only a single layer of adaptable weights. This key limitation meant that perceptron could only classify patterns that were linearly separable. Back-propagation overcomes this limitation because it can adapt two or more layers of weights, and uses a more sophisticated learning rule. The power of back-propagation lies in its ability to train hidden layers and thereby escape the restricted capabilities of perceptron-like single-layer networks.

2.2 *Formulation of Back-propagation*

In this section, a detailed derivation of Back-Propagation learning rule is presented. The back propagation network (BPN) is a layered, feed forward network that is fully interconnected by layers. There are no feedback connections. Although only three layers are used in the following formulation, more than one hidden layer is permissible. A typical BPN architecture is shown in Figure 2.1.

Let us suppose, we have a set of p vector-pairs, $(X_1,Y_1),(X_2,Y_2)$, $\ldots,(X_p,Y_p)$, which are examples of a functional mapping $Y=\phi(X) : X\epsilon R^N, Y\epsilon R^M$. If p-th input vector, $X_p=(X_{p1},X_{p2},\ldots,X_{pN})^t$, is applied to the input layer of the

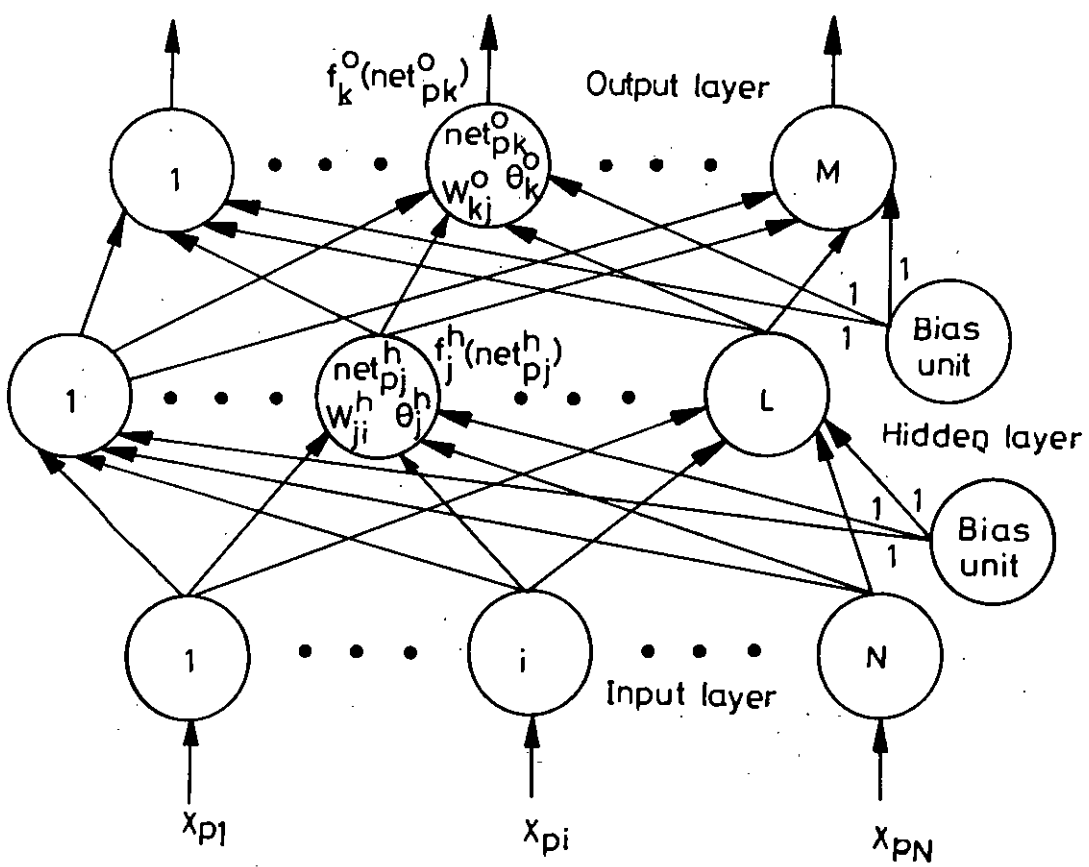


Figure 2.1 The three-layer BPN architecture.

network, the input units distribute the values to the hidden layer units and the net input to the j-th hidden units is

$$net_{pj}^{h} = \sum_{i=1}^{N} W_{ji}^{h} X_{pi} + \theta_{j}^{h} \qquad \ldots . \qquad (2\text{-}1)$$

where W^{h}_{ji} is the weight on the connection from the i-th input unit and θ^{h}_{j} is the bias term. The "h" superscript refers to quantities on the hidden layer. If it is assumed that the activation of this node is equal to the net input then the output of this node is

$$H_{pj} = f_{j}^{h}(net_{pj}^{h}) \qquad \ldots . \qquad (2\text{-}2)$$

The equations for the output nodes are similarly as follows.

$$net_{pk}^{o} = \sum_{j=1}^{L} W_{kj}^{o} H_{pj} + \theta_{k}^{o} \qquad .. \qquad (2\text{-}3) \qquad (3)$$

$$O_{pk} = f_{k}^{o}(net_{pk}^{o}) \qquad \ldots \ldots \qquad (2\text{-}4)$$

where the 'O' superscript refers to quantities on the output layer and 'k' refers to the k-th output unit.

2.2.1 *Update of Output-layer weight*

The error at a single output unit to be Y_{pk}-O_{pk}, where Y_{pk} is the desired output value of the k-th output unit. The error that is minimized by the gradient descent rule is the sum of the squares of the errors for all output

units:

$$E_p = \frac{1}{2} \sum_{k=1}^{M} (Y_{pk} - O_{pk})^2 \qquad \ldots\ldots \qquad (2\text{-}5)$$

To determine the direction in which to change the weights, the negative of the gradient of E_p, ΔE_p, with respect to the weights W_{kj} are to be calculated. Then the values of the weights can be adjusted such that the total error is reduced. Total error is defined either as the sum of the errors measured for all the patterns or the mean of all the errors. Thus the total error has any one of the following forms.

$$E = \sum_p E_p = \frac{1}{2} \sum_p \sum_{k=1}^{M} (Y_{pk} - O_{pk})^2 \qquad \ldots\ldots\ldots \qquad (2\text{-}6)$$

or

$$E = \frac{1}{p} \sum_p E_p = \frac{1}{2p} \sum_p \sum_{k=1}^{M} (Y_{pk} - O_{pk})^2 \qquad \ldots \qquad (2\text{-}7)$$

The error surface with respect to weights can have various complex forms. Figure 2.2 shows a simple example where the network has only two weights. Formulation of Back-propagation assumes that reduction of each E_p will reduce the total error E. Therefore, the change in weight W^o_{kj} for pattern 'p' is given by

$$\Delta_p W^o_{kj} \propto -\frac{\partial E_p}{\partial W^o_{kj}}$$

From Eq. (2-5),

$$\frac{\partial E_p}{\partial W_{kj}^o} = -(Y_{pk}-O_{pk})\frac{\partial O_{pk}}{\partial (net_{pk}^o)} \cdot \frac{\partial (net_{pk}^o)}{\partial W_{kj}}$$

$$= -(Y_{pk}-O_{pk})\, f_k^{o'}(net_{pk}^o)\frac{\partial (net_{pk}^o)}{\partial W_{kj}^o}$$

$$= -(Y_{pk}-O_{pk})\, f_k^{o'}(net_{pk}^o)\frac{\partial}{\partial W_{kj}^o}\left(\sum_{j=1}^{L} W_{kj}^o H_{pj} + \theta_k^o\right)$$

$$= -(Y_{pk}-O_{pk})\, f_k^{o'}(net_{pk}^o)\, H_{pj}$$

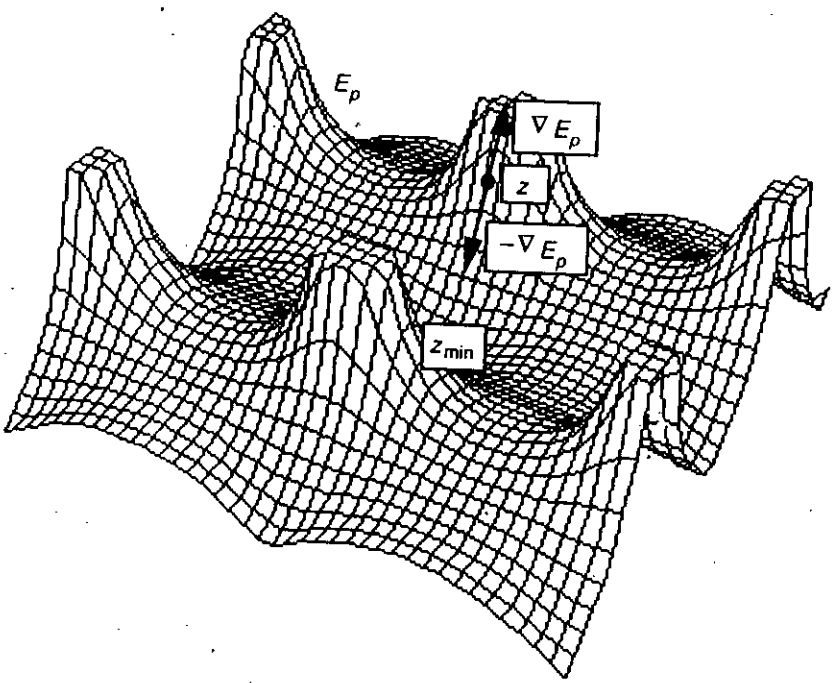


Figure 2.2 Hypothetical surface in weight space.

Therefore,

$$-\frac{\partial E_p}{\partial W^o_{kj}} = (Y_{pk} - O_{pk})\, f_k^{o'}(net^o_{pk})\, H_{pj} \qquad \ldots\ldots \qquad (2\text{-}8)$$

Therefore, the change in output layer weight W^o_{kj} for pattern 'p' is given by the following relationship.

$$\Delta_p W^o_{kj} = \eta\,(Y_{pk} - O_{pk})\, f_k^{o'}(net^o_{pk})\, H_{pj} \qquad \ldots. \qquad (2\text{-}9)$$

where the proportionality constant η is called the learning rate parameter. The outputs of hidden units and output units, i.e, H_{pj} and O_{pk} are determined by the sigmoid or logistic function which is given by

$$f(x) = \frac{1}{(1 + e^{-x})} \qquad \ldots \qquad (2\text{-}10)$$

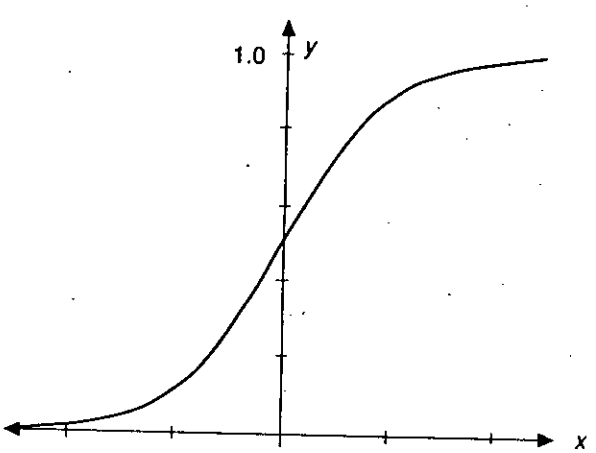


Figure 2.3 Graphical representation of sigmoid function.

Therefore

$$f_k^{o'}(net_{pk}^o) = f_k^o(net_{pk}^o)\,[1 - f_k^o(net_{pk}^o)] \ldots\ldots\ldots$$
$$= O_{pk}(1 - O_{pk})$$

Then, the Eq. (2-9) can be written as

$$\Delta_p W_{kj}^o = \eta \delta_{pk}^o H_{pj} \quad \ldots\ldots\ldots\ldots\ldots \quad (2\text{-}11)$$

Where

$$\delta_{pk}^o = (Y_{pk} - O_{pk})\, O_{pk}(1 - O_{pk}) \quad \ldots\ldots \quad (2\text{-}12)$$

is called the error signal at the output layer.

2.2.2 *Update of Hidden-layer Weight*

To extend the above weight update procedure in the hidden layer it is necessary to determine a measure of the error of the outputs of the hidden layer units. Intuitively, the total error, E must somehow be related to the output values (H_{pj}) on the hidden layer and H_{pj} depends on the weighs on the hidden layer. This fact can be exploited to calculate the gradient of E_p with respect to the hidden layer weights. For pattern 'p', the weights in the hidden layer can be updated as

$$\Delta_p W^h_{ji} \propto - \frac{\partial E_p}{\partial W^h_{ji}} \quad \ldots\ldots$$

Now, from Eq. (2-5)

$$\begin{aligned}
\frac{\partial E_p}{\partial W^h_{ji}} &= \frac{1}{2}\sum_k \frac{\partial}{W^h_{ji}} (Y_{pk} - O_{pk})^2 \\
&= -\sum_k (Y_{pk} - O_{pk}) \frac{\partial O_{pk}}{\partial (net^o_{pk})} \cdot \frac{\partial net^o_{pk}}{\partial H_{pj}} \cdot \frac{\partial H_{pj}}{\partial (net^h_{pj})} \cdot \frac{\partial net^h_{pj}}{\partial W^h_{ji}} \\
&= -\sum_k (Y_{pk} - O_{pk})\, f^{o'}_k (net^o_{pk})\, W^o_{kj} f^{h'}_j (net^h_{pj})\, X_{pi} \\
&= -f^{h'}_j (net^h_{pj})\, X_{pi} \sum_k (Y_{pk} - O_{pk})\, f^{o'}_k (net^o_{pk})\, W^o_{kj} \\
&= -f^{h'}_j (net^h_{pj})\, X_{pi} \sum_k \delta^o_{pk} W^o_{kj} \qquad \ldots
\end{aligned}$$

Therefore, change in weight W^h_{ji} for pattern 'p' is governed by the following relationship.

$$\Delta_p W^h_{ji} = \eta f^{h'}_j (net^h_{pj})\, X_{pi} \sum_k \delta^o_{pk} W^o_{kj} \quad \ldots\ldots \qquad (2\text{-}13)$$

Every weight update on the hidden layer depends on all the error terms, δ^o_{pk}, on the output layer. The known errors on the output layer are propagated back to the hidden layer to determine the appropriate weight changes on that layer. By defining a hidden layer error signal term δ^h_{pj} as

$$\delta^h_{pj} = f^{h'}_j (net^h_{pj}) \sum_k \delta^o_{pk} W^o_{kj} \qquad \ldots \quad (2\text{-}14)$$

the weight update equation on hidden layer become analogous to those for the output layer.

Thus training a set of patterns by Back-propagation involves two phases. As a pattern is presented to the network, it propagates through the network and produces outputs at the output layer - this is called propagation phase. Error is determined by comparing the actual output with the target value and propagated backward through the network to make weight modification - this is called back-propagation phase. After the complete set of patterns is presented to the network and necessary weight modification is done each time, a training cycle or epoch is said to be completed. Training continues until the error reduces to some prefixed value and this usually takes a number of training cycles.

2.3 *Limitations Standard Back-propagation algorithm*

Though back propagation learning algorithm is one of the most popular learning schemes to date for multi-layer networks, it does have several inevitable limitations. Some of them are as follows.

(1) The number of training cycles required for the sum-squared error to be reduced to an acceptable value is usually large in most cases. The reasons for this problem has been discussed in detail in the next section.

(2) It sometimes gets stuck in local minima. In that case extremely long training cycles are necessary for convergence.

(3) The architecture of Back-propagation algorithm is fixed. There is no provision for changing the number of hidden-layer units during training.

(4) The Back-propagation algorithm is incapable of incremental learning

2.4 *Reasons for slow convergence*

While steepest descent can be an efficient method for obtaining the weight values that minimizes an error measure, error surfaces frequently possess properties that make this procedure slow to converge. There are at least three reasons for this slow rate of convergence. The first two reasons involve the magnitude of the components of the gradient vector and the direction of the gradient vector. The third one is the procedural problem. The reasons are:

(1) The value of η, which modulates the step size, is sensitive to the local shape of the hyper-dimensional terrain which is being traversed in the optimization. If a steep v-shaped valley is being followed (particularly if that

valley has twist and turns), too large a value of η will cause steps to bounce between the two opposite sides of the valley rather than following the contour of its bottom. On the other hand, too small a value of η will prevent the system from making reasonable progress across a long flat slope.

(2) A step (a change of weight and activations based on the corrections produced by the back-propagation) that reduces the error with respect to one pattern will not, in general, produce a network with reduced error with respect to all the other patterns which the system is to learn. Such a step may misdirect the optimization path which means that the negative gradient vector may not point directly towards the minimum of the error surface. This is illustrated in Figure 2.4 for a two dimensional weight space.

The error surface is drawn topographically using contours to represent regions of equal error. The minimum of the error surface is represented by the back dot in the center of the contours. The current weight vector is given by w(t). Since at the current point in weight space the error surface is steeper along the w_2 dimension than the w_1 dimension, the derivative of w_2 is greater than the derivative of w_1. In general, for the direction of the negative gradient vector to equal the direction of the minimum of the error surface for all points in weight space, the contours that represent regions of equal error must be circular.

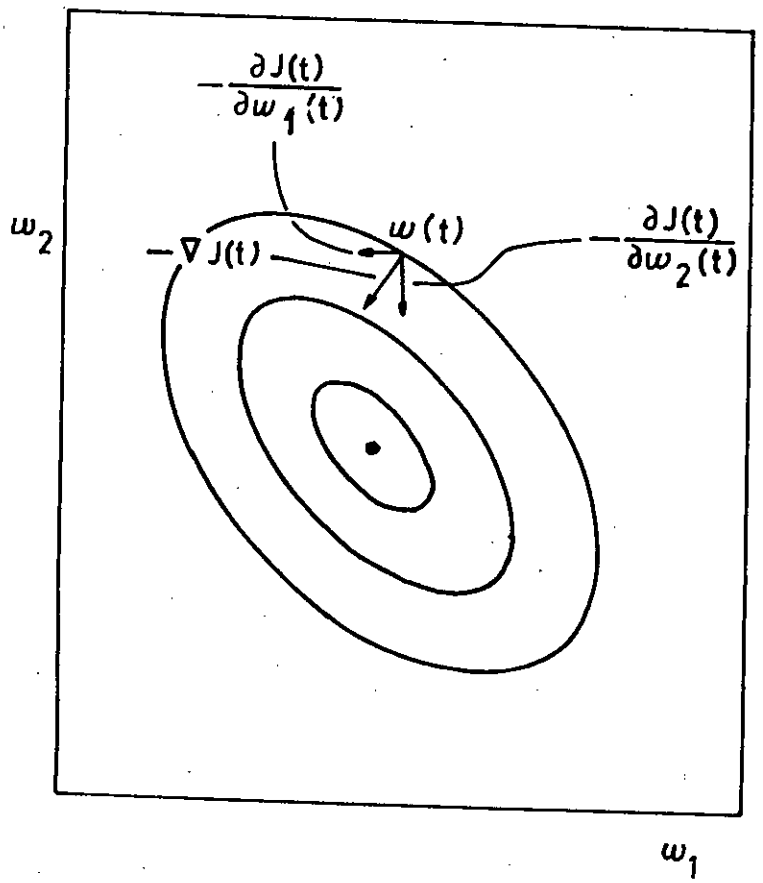


Figure 2.4 Error surface over two-dimensional weight space.

3) the weights are adjusted after every trial regardless of the benefit of such an adjustment. In reality, however, many consecutive data points contain redundant information. Updating without fresh information not only is inefficient but also can be detrimental due to potential inconsistency. Furthermore, the processing time required by updating may also be

unnecessarily long, thus handicaps the implementation. Also an inappropriately chosen initial weight set could result in undesirably long training period.

2.5 *Summary*

In this chapter, a detail derivation of standard Back-Propagation algorithm is presented and several limitations of this algorithm, especially, slow convergence and its reasons have been discussed. In the next chapter, some significant works on accelerating the convergence speed are presented.

CHAPTER 3

PREVIOUS WORKS ON ACCELERATED CONVERGENCE

INTRODUCTION

SURVEY OF PREVIOUS WORKS

SUMMARY

3.1 *Introduction*

The greatest single obstacle to the widespread use of back-propagation learning algorithm in real-world applications is its slow speed in convergence and it scales up poorly as tasks become larger and more complex. Even on relatively simple problems, standard back-propagation often requires a lengthy training process in which the complete set of training examples is processed hundreds or thousands of times. This limits its use to investigating rather small networks with only a few thousand trainable weights. Some problems of real-world importance can be tackled using networks of this size, but most of the tasks for which connectionist technology might be appropriate are much too large and complex to be handled by the current training algorithms. Much of the efforts in neural networks research is devoted to developing faster convergent algorithms. In the following sections, some of the earlier works to accelerate Back-propagation convergence is presented.

3.2 *Survey of Previous Works*

After considering the reasons for the possible slow rate of convergence of the steepest descent procedure as mentioned in chapter 2, several methods have been proposed by several researchers.

In [5], Vogl et.al. modified the standard Back-propagation algorithm on the basis of experience gained with other non-linear optimization algorithms. The following modifications were proposed.

1. After each pattern is presented, all weight changes are calculated as usual through back-propagation, but these changes are not immediately

applied. Instead, the changes for each weight are summed over all of the input patterns and the sum is applied to modify the weight after each iteration over all the patterns. Thus updating is modified as follows:

$$\Delta w_{jk}(m+1) = \eta \sum_{p} (\delta_{pj} . O_{pk}) + \alpha . \Delta w_{jk}(m) \quad .... \quad (3-1)$$

where m represents the iteration number rather than the presentation number.

2. The learning rate η is varied during training. η is increased or decreased by certain factor depending on whether preceding updating decreased or increased the total error.

This modification requires two additional parameters to choose. These two additional parameters and initial values of η must be carefully chosen. Fast increase of some parameters may cause unstable behavior in the learning process. Moreover, with this modification, early stage of learning seems to very slow. However, this modification has not been adequately investigated and has not gained wide acceptance.

In [8], Jacob proposed heuristics that provide guidelines for achieving faster rate of convergence than steepest descent techniques. These heuristics take four points into consideration.

First, every weight should have its own individual learning rate. The argument is that the step size appropriate for one weight is not necessarily appropriate for other weights.

Second, every learning rate should be allowed to vary over time. It is common for error surfaces to possess different properties along different regions of a single parameter dimension. In order to take appropriate step sizes as the weight varies the learning rate will also need to vary.

Third, when the derivative of error with respect to a weight possesses the same sign for several consecutive iterations its learning rate should be

increased. It is frequently the case where the error surface at the current point in weight space along that weight dimension possesses a small curvature along that weight and therefore, continues to slope in the same direction for some significant distance. Incrementing learning rate for this weight can reduce the number of time steps required to traverse this distance.

Fourth, when the sign of the derivative for a weight alternates for several consecutive time steps, its learning rate should be decreased. It is frequently the case where the error surface along that weight dimension possesses a high curvature and therefore, the slope of this area of the error surface may quickly change sign. In order to prevent this weight from oscillating, this value should be adjusted by a smaller amount.

Based on the above consideration, Jacobs suggested individual learning rate for each weight and updating of this learning rate on an estimate of the curvature of error surface at the current point along that weight. A learning rate updating rule as suggested by Jacobs in [8] is as follows

$$\Delta\eta_{ji}(t) = \gamma \frac{\partial E(t)}{\partial W_{ji}(t)} \cdot \frac{\partial E(t-1)}{\partial W_{ji}(t-1)} \quad \cdots \quad (3-2)$$

where η_{ji} is the learning rate of weight W_{ji}.

Simulation results provided by Jacobs in [8] show improvement of convergence. However, this modification makes learning rate much more complex and doubles the number of parameters to be adjusted. Improvement in terms of training cycles required for convergence does not necessarily imply less computational cost. A comparison in terms of actual computational cost has not been investigated. Moreover, there exist error surfaces where use of these heuristics may not result in the desired effects [8].

S.C. Huang and Y.F. Huang [12] suggested that weights should not be

adjusted after every trail regardless of benefit. There are two criteria involved in the selection process. One is concerned with initial condition while the other one chooses an appropriate sequence of presentation of training data.

In [11], Samad argues that the effectiveness of weight update in standard Back-propagation is reduced due to the fact that, as learning proceeds, the output of a unit in response to an input vector deviates from the value used in weight update. The reason is that, to update a particular weight Back-propagation takes gradient on an error surface assuming that all the other weights remains stationary. But this is not the case in real implementation of the algorithm. Thus the error surface appears to be changing every time the weights are updated. To compensate this change, Samad proposed to use an expected value of the unit whose associated weight is to update by the following rule:

$$\Delta W_{ji}=\eta\delta_j(O_i+\beta\delta_i) \qquad \ldots\ldots\ldots \qquad (3-3)$$

Where ß is a constant and δ's are the error signal of units.

This modification showed faster convergence in some cases. However, performance of this modification is highly dependent on η, and for some range of η, this modification does not show any speed up. However, proper choice of the value of ß is also problem-specific.

In [10], Holt et. al. proposed a log-likelihood cost function as an alternate to sum-squared error used in standard Back-propagation. This cost function is defined as

$$E=-\sum_p\sum_j t_{pj}\ Log\ o_{pj}+(1-t_{pj})\ Log(1-o_{pj}) \qquad .. \quad (3-4)$$

Updating of weights is done using the gradient of this error surface. This

gives the following form of error signal at the output layer.

$$\delta^{o}_{pk} = (Y_{pk} - O_{pk}) \qquad \ldots \qquad (3\text{-}5)$$

Comparison with equation (2-10) shows the absence of the term $O_{pk}(1-O_{pk})$ in equation (3-5). In fact the term $O_{pk}(1-O_{pk})$ is a bell-shaped function having a maximum value of 0.25. This actually scales down the effective step size in standard Back-propagation. Thus use of Loglikelihood cost function [equation (3-5)] makes early stage of learning considerably faster. However, convergence speed does not show significant improvement at the final stage of learning. One limitation of this new cost function is that it cannot be used in applications where linear units are needed at the output layer.

3.3 *Summary*

In this chapter, some of the previous works on accelerated convergence is presented. From these works, it is obvious that acceleration can be achieved either by some techniques and/or by defining new cost function. Whatever modification is done it should not increase the computational complexity or computational cost per iteration. In the next chapter, a new cost function is proposed which adapts the learning rate parameter during training in a simple manner.

CHAPTER–4

PROPOSED ACCELERATION TECHNIQUE

PROPOSED COST FUNCTION

WEIGHT UPDATING USING PROPOSED COST FUNCTION

FURTHER MODIFICATION

SUMMARY

4.1 *Proposed Cost Function*

Based on the analysis of the reasons of slow convergence in Back-propagation algorithm presented in the previous chapter, a new cost function is proposed with the intention of achieving accelerated convergence. With this cost function, the effective learning rate parameter is made to vary as a function of error measure. This function is defined as

$$E^t = \sum_p E_p^t = \sum_p e^{\beta E_p} \qquad \ldots\ldots\ldots\ldots \quad (4\text{-}1)$$

where E^t is the total error and E^t_p is the measure of error at the output layer for pattern 'p'. E_p as defined in the previous chapter has the following form.

$$E_p = \frac{1}{2}\sum_{k=1}^{M} (Y_{pk} - O_{pk})^2 \qquad \ldots\ldots\ldots \quad (4\text{-}2)$$

where Y_{pk} and O_{pk} are the target and actual output of unit 'k' for pattern 'p'.

4.2 *Weight Updating Using Proposed Cost Function*

Change in an output layer weight W^o_{kj} for pattern 'p' is governed by

$$\Delta_p W^o_{kj} \propto - \frac{\partial E_p^t}{\partial W^o_{kj}}$$

Now from Equation (4-1)

$$\frac{\partial E_p^t}{\partial W_{kj}^o} = \beta e^{\beta E_p} \frac{\partial E_p}{\partial W_{kj}^o}$$
$$= \beta e^{\beta E_p}[-(Y_{pk}-O_{pk}) f_k^{o'}(net_{pk}^o) H_{pj}]$$
$$= -\beta e^{\beta E_p}(Y_{pk}-O_{pk}) f_k^{o'}(net_{pk}^o) H_{pj} \quad \ldots . (4-3)$$

A close observation of equation (4-3) shows that the term $\beta e^{\beta E_p}$ has to be calculated each time a pattern is presented. Since calculation of this term involves calculation of E_p for every weight updating, several multiplications and exponential computations. The computational overload using the proposed cost function can be reduced if the term $\beta e^{\beta E_p}$ is replaced or approximated by $\beta e^{\beta E}$ where E is defined in equation (2-7). Then this term needs to be calculated only once for each training cycle and can be used for presentation of each pattern. Only then a direct comparison of convergence speed of this modification with standard Back-propagation in terms of training cycles is possible.

For sigmoid activation function the change in weight for each pattern 'p' becomes

$$\Delta_p W_{kj}^o = \eta\beta e^{\beta E}(Y_{pk} - O_{pk}) O_{pk}(1-O_{pk}) H_{pj}$$
$$= \eta\beta e^{\beta E} \delta_{pk}^o H_{pj}$$
$$= \eta_{eff}\, \delta_{pk}^o H_{pj} \quad .. \quad (4-4)$$

where,

$$\eta_{eff} = \eta\beta e^{\beta E} \quad ..... \quad (4-5)$$

and

$$\delta_{pk}^o = (Y_{pk} - O_{pk}) O_{pk}(1 - O_{pk}) \quad . \quad (4-6)$$

where δ^o_{pk}, H_{pj} are the error signal at output unit 'k' and output of hidden unit 'j' respectively. η_{eff} is the effective learning rate parameter.

It is worth noting here that now the effective learning rate parameter η_{eff} is the learning rate parameter in standard Back-propagation η multiplied by a factor which is an exponential function of the error measure at output layer. Thus it will vary the learning rate as the learning proceeds. This will keep the learning rate high at the beginning and low at the final stage of learning, and adapt according to the error measure as necessary.

Similarly change in hidden layer weight W_{ji}^{h} for each pattern is given by

$$\Delta_p W_{ji}^{h} = \eta_{eff} \delta_{pj}^{h} X_{pi} \quad \ldots\ldots\ldots \quad (4\text{-}7)$$

and

$$\delta_{pj}^{h} = H_{pj}(1 - H_{pj}) \sum_{k} \delta_{pk}^{o} W_{kj}^{o} \quad \ldots \quad (4\text{-}8)$$

where δ^{h}_{pj} and X_{pi} are the error signal of hidden unit and input respectively.

4.2.1 *Further Modification*

As stated in the previous chapter, the modification proposed by Halt [10] using Log-likelihood cost function eliminates the term $O_{pk}(1-O_{pk})$ in the output-layer weight update equation. It would be interesting to see how the elimination of the above term in the weight update equation [equation (4-4)] using proposed cost function affect the convergence speed. Thus further modification of update weight which has also been studied has the form

$$\Delta_p W_{kj}^{o} = \eta \beta e^{\beta E} (Y_{pk} - O_{pk}) H_{pj} \quad \ldots\ldots\ldots \quad (4\text{-}9)$$

4.3 *Summary*

In this chapter, a new cost function is proposed and formulation of the weight update equations with this new cost function is presented. Since there is no analytical method of evaluating the convergence speed of BP algorithm, the effectiveness of the proposed modification must be verified by extensive simulation results on a variety of problems. In the next chapter, simulation results for a number of problems investigating the convergence behavior and speed is presented.

CHAPTER-5

SIMULATION RESULTS

INTRODUCTION

XOR PROBLEM

CHARACTER RECOGNITION PROBLEM

FUNCTION APPROXIMATION PROBLEM

SUMMARY

5.1 *Introduction*

In order to investigate the convergence behavior and speed of the modification proposed in the preceding chapter, simulations are carried out with different types of problems. Since no analytical techniques are available to study the convergence of Back-propagation algorithm, simulation with different problems and comparison with standard Back-propagation is the usually adopted means to evaluate the effectiveness of a modification. In the present work, investigation has been done with three different problems. First one is the XOR problem which is treated as a benchmark problem in the neural network literature and has been used to test the modifications proposed in [5]-[12]. Second one is a character recognition problem. This problem is chosen because neural network is well suited for applications in recognition system. Third one is a function approximation problem. In this case, both the inputs and targets are analog values instead of binary values used in the former ones.

In the following sections, results of the simulation with the proposed modification and performance comparison with the standard Back-propagation are presented.

5.2 *XOR Problem*

Exclusive-OR problem is particularly important because of its nonlinear separability and has been studied extensively in neural network literature. Rumelhart [4] and others have taken this problem as a benchmark one and investigated in details using not only Back-propagation and its derivatives but also other algorithms and architectures. Although the problem is of very

small size having only two inputs and single output as shown in Table 5.1, it takes comparatively large number of training cycles compared to other problems of similar size.

Table 5.1 Input-Output patterns for XOR problem.

Input		Output
0	0	0
0	1	1
1	0	1
1	1	0

A 2-2-1 (two input, two hidden and one output units) network and a 2-3-1 (two input, three hidden and one output units) network were trained using both standard back-propagation and the proposed modification described in the previous chapter. Investigation was done with different values of η, ß (in case of proposed modification only), and training was continued for different values of error limit. Simulation results for three different values of η with different ß and error limit are summarized in Tables 5.2 and 5.3. In each case ß was varied from 1.0 upto 3.0 or 4.0. It should be mentioned that total training cycles required to reduce the error to a certain limit is also dependent on the initial weights chosen. For this reason, 5 trails with different initial weights were carried out and the average training cycles required is presented in Tables 5.2 and 5.3. A graphical representation of the results are shown in Figures 5.1 and 5.2.

Simulation results with XOR show that the use of the proposed cost function greatly accelerates the convergence speed within certain range of ß. In case of XOR, this range is found to be within ß=1.0 to 3.0 or 4.0. A typical convergence behavior within this range is shown in Figures 5.3 and 5.4 for 2 and 3 hidden units respectively. These Figures show that error

Table 5.2 Comparison of training cycles required to realize XOR problem with a 2-2-1 network using standard Back-propagation and proposed modification. (a) For η =0.20 (b) For η =0.40 (c) For η =0.60

(a)

Error Limit	η=0.20							
	Standard BP	Modified B.P						
		β=1.0	β=1.50	β=2.0	β=2.5	β=3.0	β=3.5	β=4.0
0.01	3700	2992	1813	1252	936	755	622	533
0.001	16010	15271	9990	7377	5830	4846	4120	3586
0.0005	28668	27922	18427	13702	10892	9080	7750	6758

(b)

Error Limit	η=0.40							
	Standard BP	Modified B.P						
		β=1.0	β=1.50	β=2.0	β=2.5	β=3.0	β=3.5	β=4.0
0.01	1826	1481	900	630	480	452	379	421
0.001	7980	7607	4990	3691	2924	2456	2119	1922
0.0005	14312	13948	9210	6858	5458	4555	3934	3503

(c)

Error Limit	η=0.60							
	Standard BP	Modified B.P						
		β=1.0	β=1.25	β=1.50	β=1.75	β=2.0	β=2.25	β=3.0
0.01	1207	982	751	602	500	427	375	323
0.001	5310	5076	4025	3327	2832	2467	2186	1677
0.0005	9535	9297	7402	6144	5249	4581	4063	3091

Table 5.3 Comparison of training cycles required to realize XOR problem with a 2-3-1 network using standard Back-propagation and proposed modification. (a) For $\eta = 0.20$ (b) For $\eta = 0.40$ (c) For $\eta = 0.60$

(a)

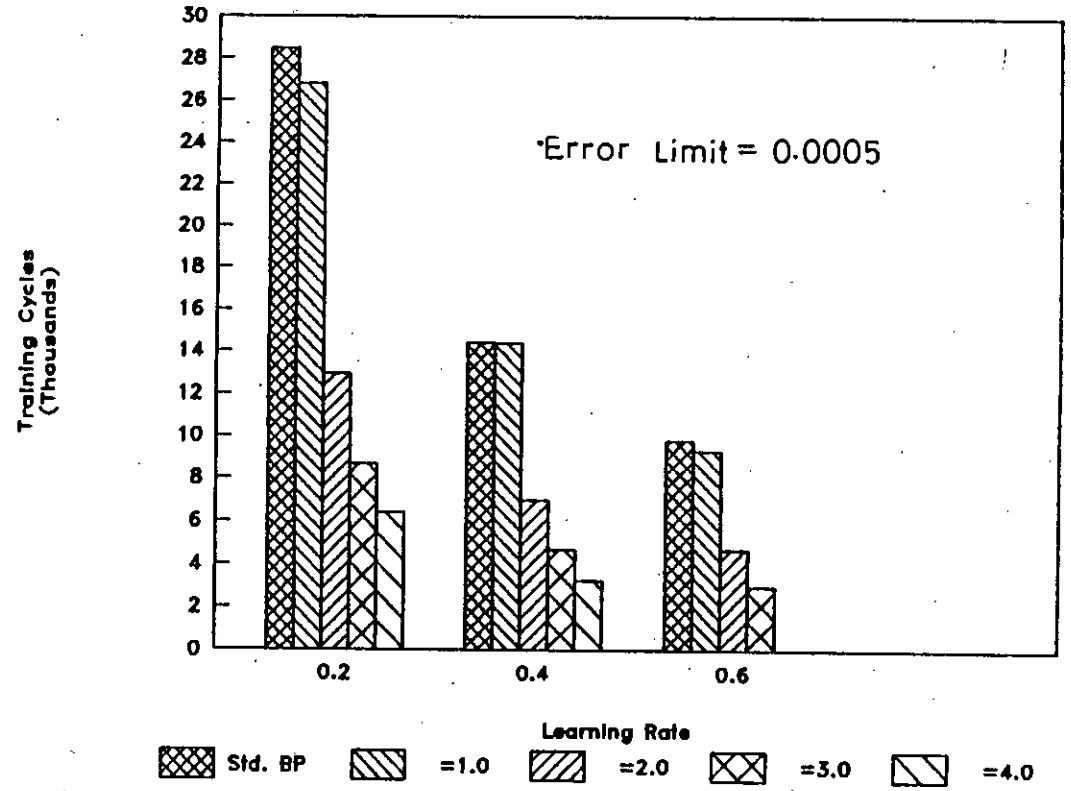
Error Limit	η=0.20							
	Standard BP	Modified B.P						
		β =1.0	β=1.50	β=2.0	β=2.5	β=3.0	β=3.5	β=4.0
0.01	5186	3735	2112	1420	1063	860	715	648
0.001	16770	15211	9731	7132	5675	4726	3970	3497
0.0005	28479	26804	17460	12937	10360	8656	7287	6407

(b)

Error Limit	η=0.40							
	Standard BP	Modified B.P						
		β=1.0	β=1.50	β=2.0	β=2.5	β=3.0	β=3.5	β=4.0
0.01	2424	1900	1110	772	595	498	393	357
0.001	8374	8071	5076	3841	3060	2570	2062	1781
0.0005	14426	14381	9102	6969	5564	4665	3802	3251

(b)

Error Limit	η=0.60							
	Standard BP	Modified B.P						
		β=1.0	β=1.25	β=1.50	β=2.0	β=2.5	β=3.0	β=3.5
0.01	1445	1073	802	638	482	405	329	324
0.001	5572	5143	4026	3325	2548	2050	1632	1492
0.0005	9796	9330	7345	6097	4670	3745	2955	2693



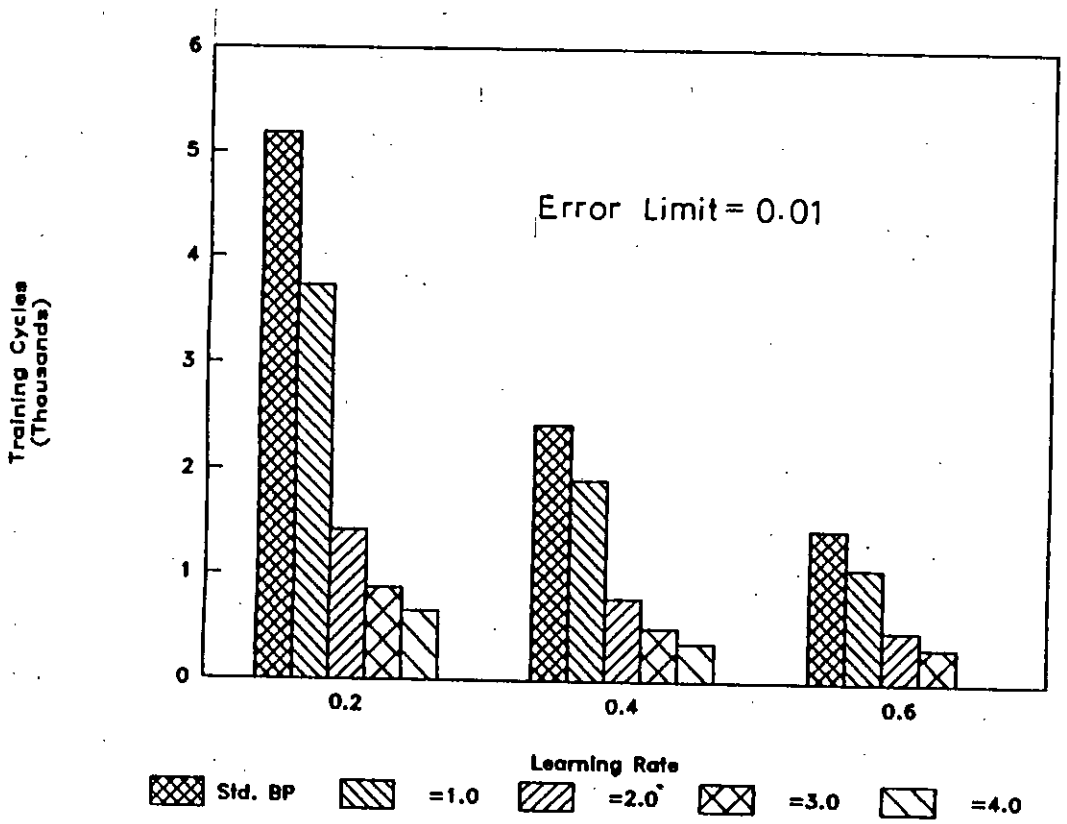
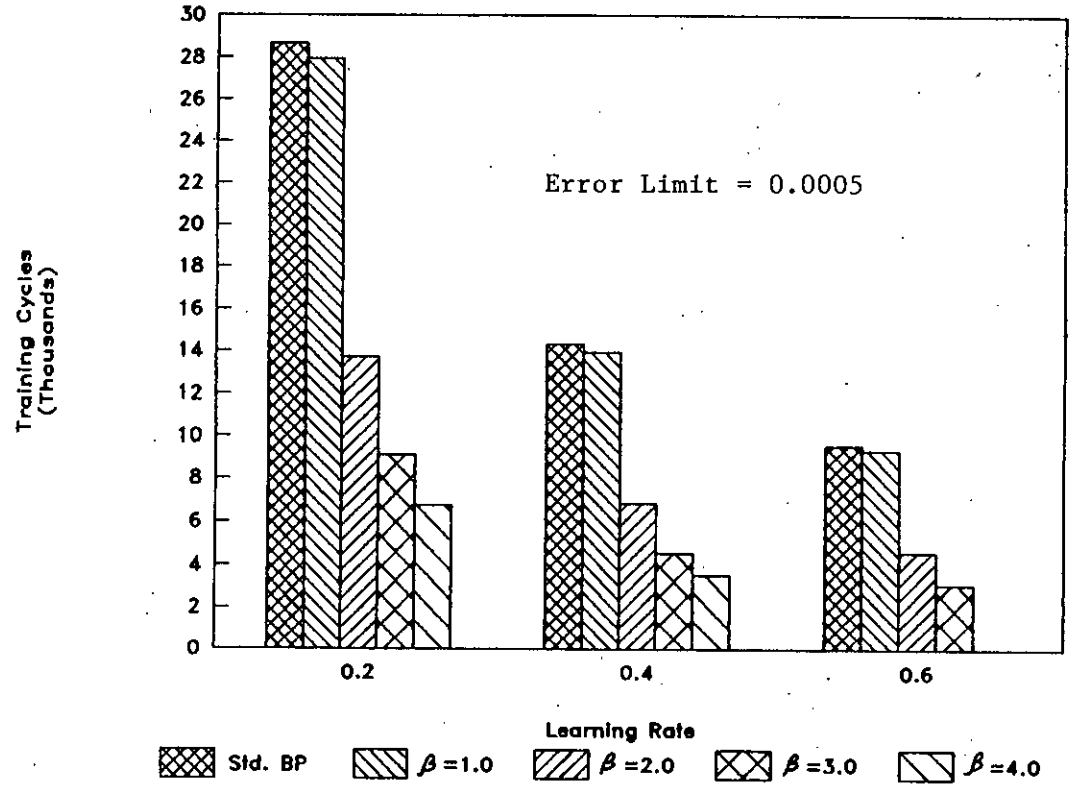


Figure 5.2 Comparison of convergence speed between standard Back-propagation and the proposed modification to learn XOR problem with a 2-3-1 network for different values of η and ß.



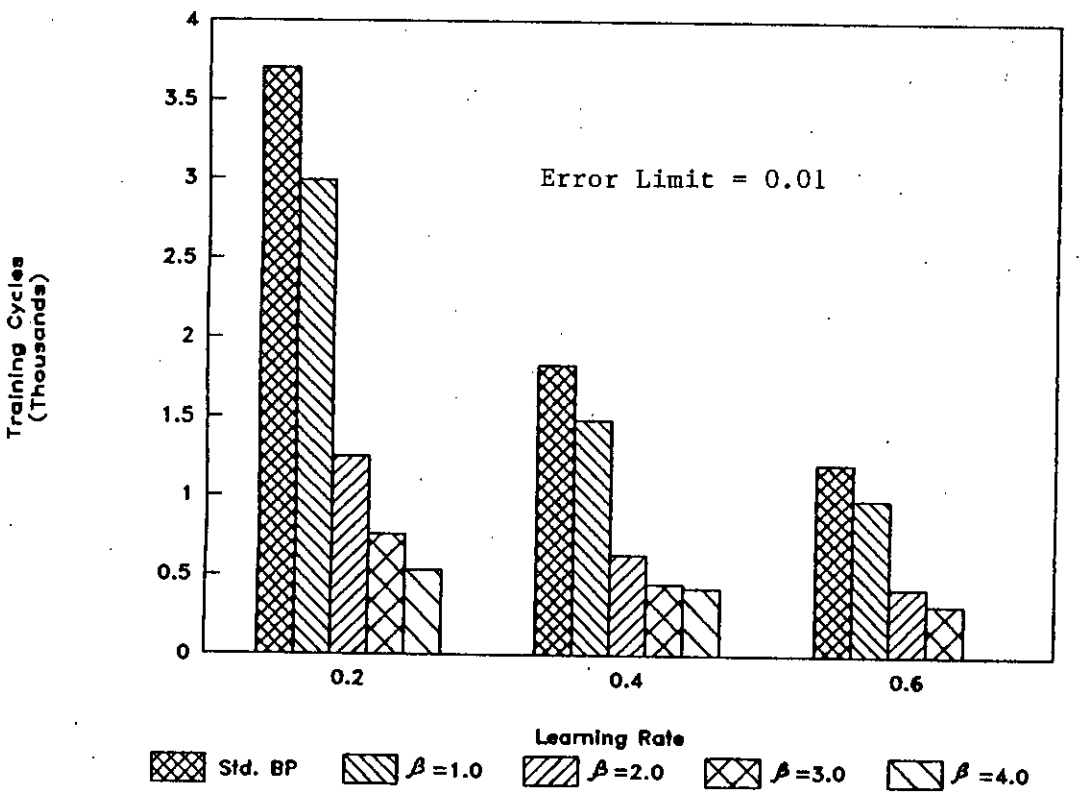


Figure 5.1 Comparison of convergence speed between standard Back-propagation and the proposed modification to learn XOR problem with a 2-2-1 network for different values of η and ß.

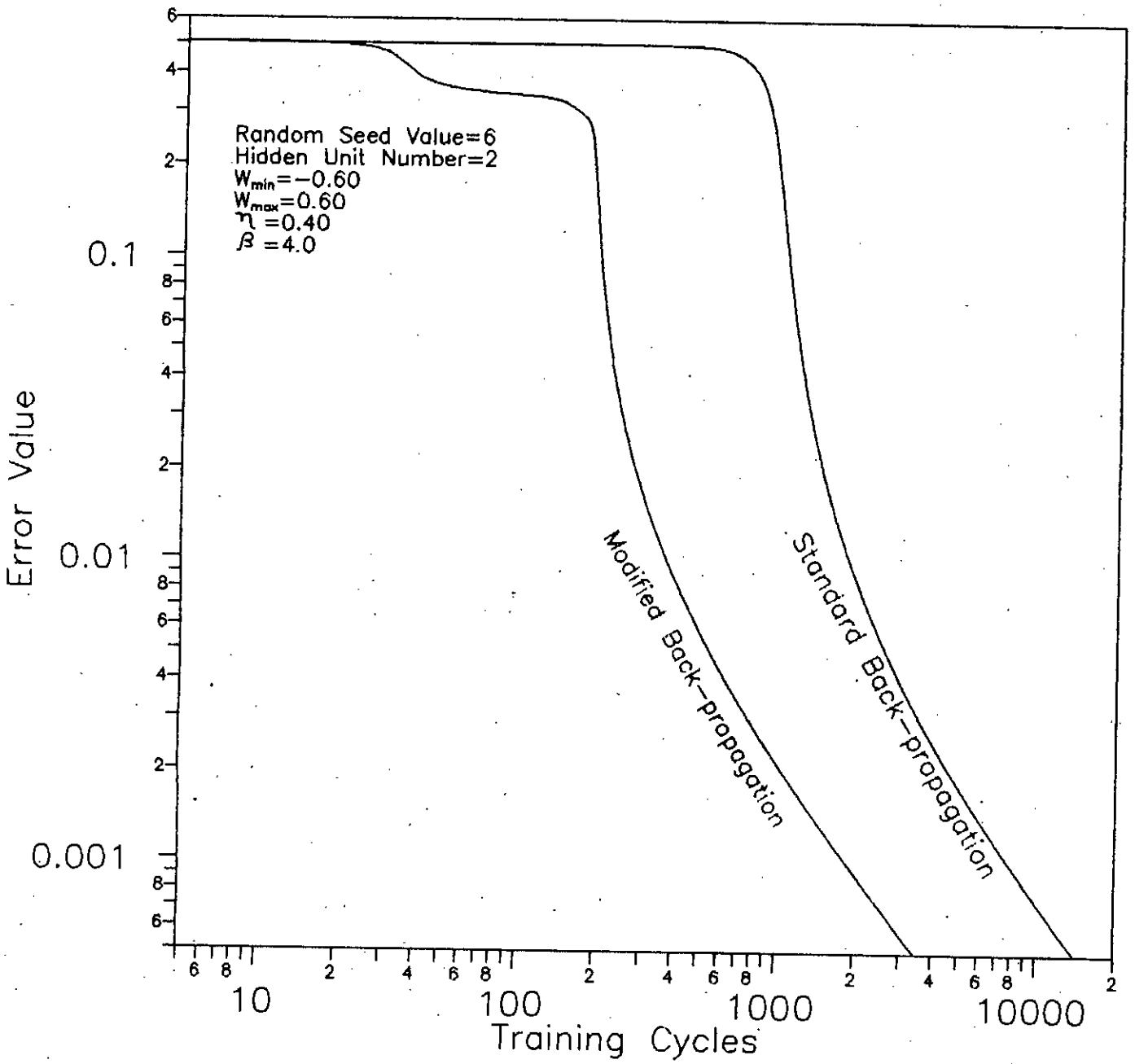


Figure 5.3 Typical learning characteristic for XOR problem with 2 hidden units using standard Back-propagation and Proposed modification.

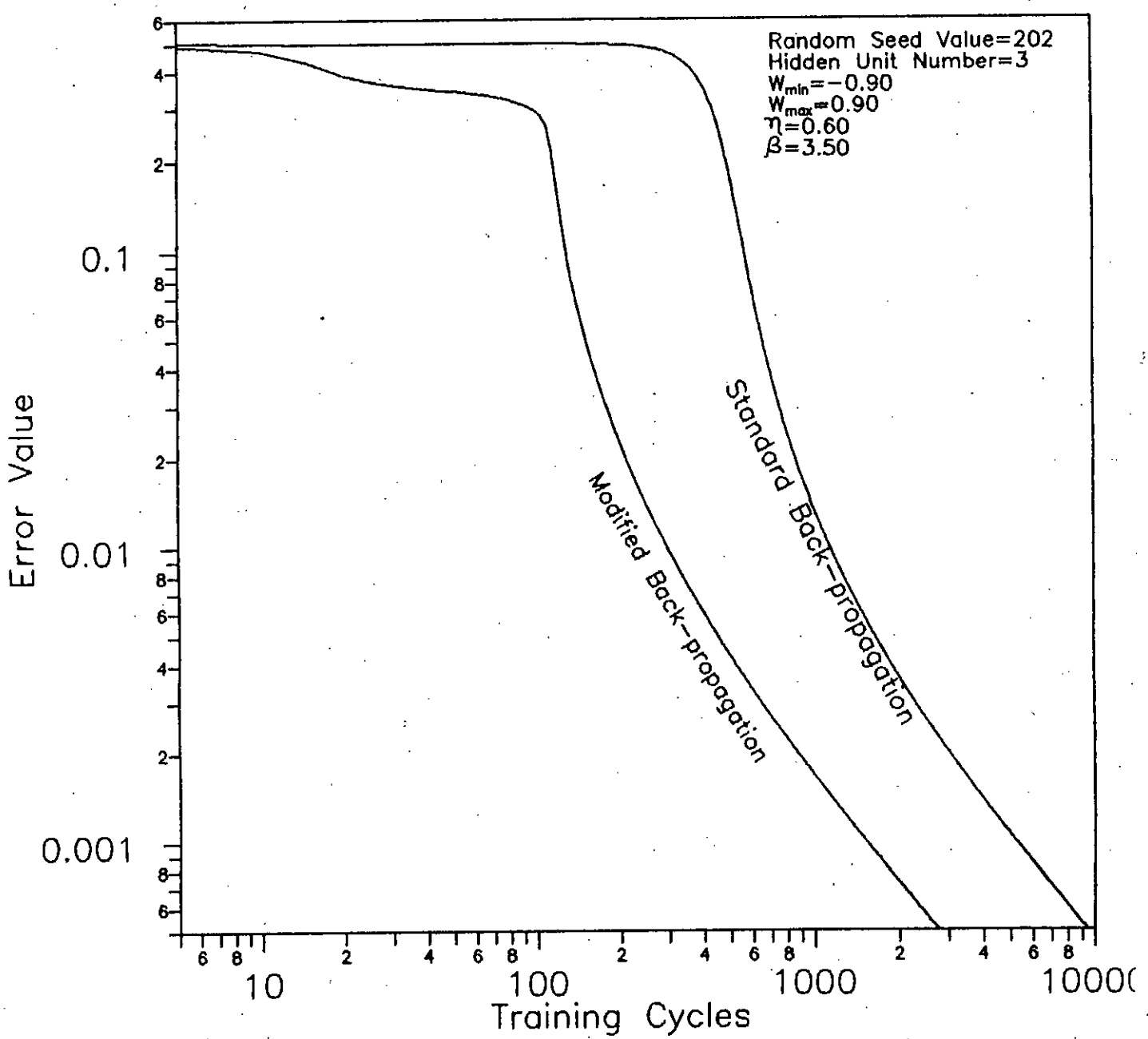


Figure 5.4 Typical learning characteristic for XOR problem with 3 hidden units using standard Back-propagation and Proposed modification.

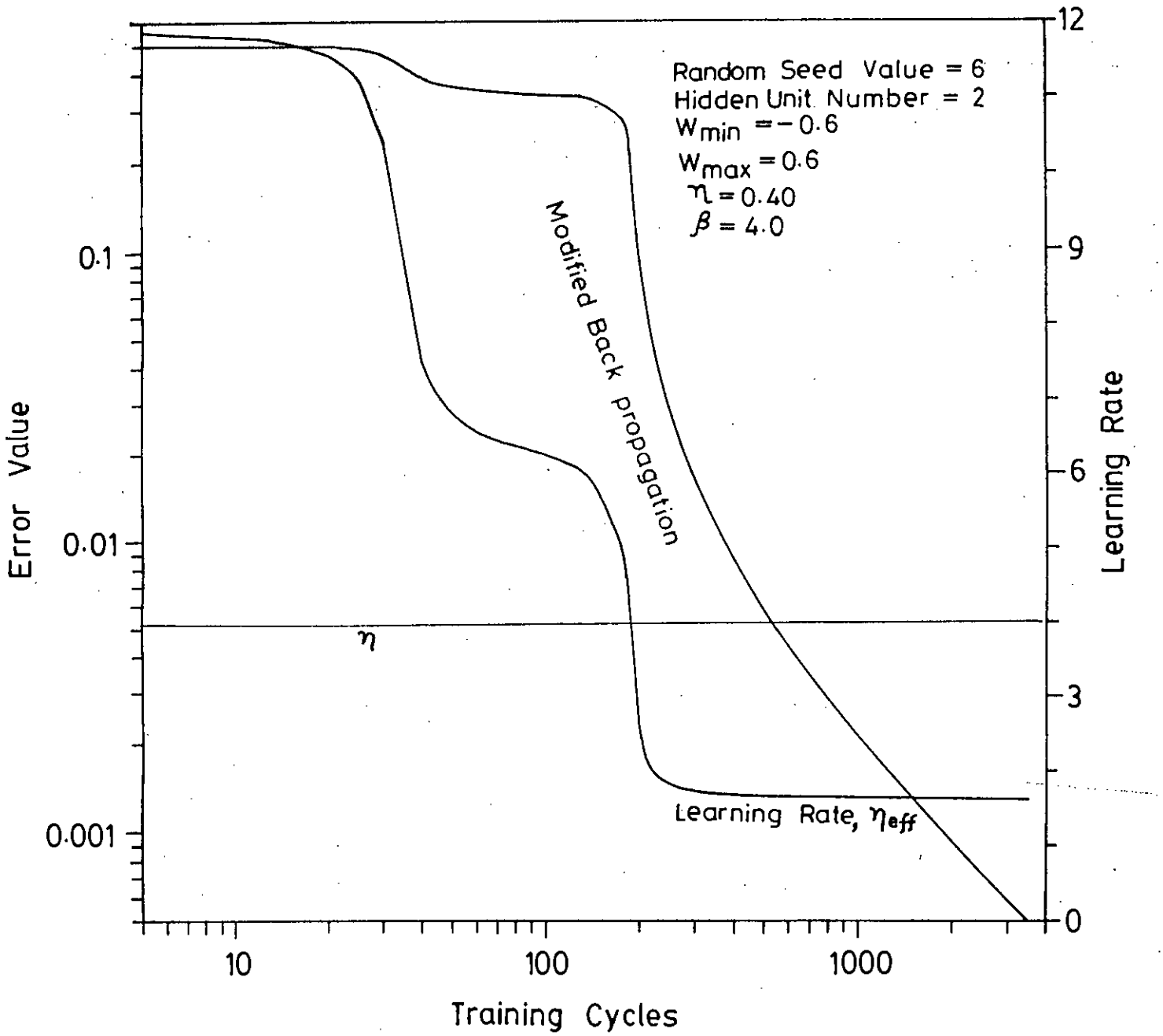


Figure 5.5 Variation of effective learning rate η_{eff} with error as learning proceeds in case of XOR problem for 2-2-1 network.

Table 5.4 Comparison of training cycles required to realize XOR problem with a 2-2-1 network using standard Back-propagation with Log-likelihood (LL) cost function and modification as per Eq. (4-9). (a) For η=0.20 (b) For η=0.40 (c) For η=0.60

(a)

Error Limit	η=0.20							
	BP (LL)	BP modified as per Eq. (4-9)						
		β=0.75	β =1.0	β=1.25	β=1.5	β=1.75	β=2.0	β=2.50
0.01	630	669	475	356	282	232	196	148
0.001	1039	1224	882	682	551	462	396	306
0.0005	1272	1532	1115	868	708	595	513	400

(b)

Error Limit	η=0.40							
	BP (LL)	BP modified as per Eq. (4-9)						
		β=0.75	β=1.0	β=1.25	β=1.5	β=1.75	β=1.9	β=2.0
0.01	311	327	241	184	150	125	115	108
0.001	515	595	442	345	282	239	217	205
0.0005	631	747	559	440	361	306	277	264

(c)

Error Limit	η=0.60							
	BP (LL)	BP modified as per Eq. (4-9)						
		β=0.25	β=0.50	β=1.0	β=1.25	β=1.3	β=1.4	β=1.5
0.01	220	786	363	165	130	126	116	107
0.001	345	1330	632	298	235	226	209	194
0.0005	422	1640	787	377	298	285	264	246

starts to reduce drastically after it has reached nearly 0.5 both in standard Back-propagation and modification. But in case of modification, it starts after 200 cycles whereas in standard Back-propagation after 1500 cycles. Similar trend is observed for other values of η and ß also. Figure 5.5 shows how the effective learning rate η_{eff} varies for the learning shown in Figure 5.4. This shows that the adaptability of η_{eff} with changing error results in faster learning. Starting with high effective learning rate it finally reduces to smaller values. This is quite justified because at the latter stage of learning when error is of small value, a large step size can retard the learning process, even cause oscillation. By varying the learning rate as required, the learning was significantly reduced.

XOR problem was also studied with Log-likelihood cost function proposed by Halt [10] and the modification suggested in Eq. (4-9). It should be mentioned here that the modification suggested in Eq. (4-9) follows similar argument presented by Halt. Although the algorithm in [10] minimizes Log-likelihood error defined by Eq. (3-4), a comparison of convergence speed was made on the basis of reduced sum-squared error. Simulation results are presented in Table 5.4 for different values of η and ß Range of ß that yields accelerated convergence is found roughly within $1.0 \sim 2.0$. This shows a reduced range of ß compared to the results presented before. The reason is that, in this case absence of the term $O_{pk}(1-O_{pk})$ in Eq. (4-9) further increases the effective step size. However, in this case also modification shows significant acceleration in convergence speed.

5.3 *Character Recognition Problem*

The second problem investigated was a character recognition one. A Back-propagation network was trained to learn ten alphanumeric digits

(0 9) shown in figure 5.6. Each digit is a 8X8 pixel. Each black pixel is represented by '1' and white pixel by '0'. Target signal at the output layer is a locally represented one, i.e., only one output unit representing a signal pattern is active and the rest are zero. Thus the network has 64 inputs, and ten output units are needed to represent ten categories. A network with 64 input units, 10 output units and 5 hidden units (64-5-10) and a (64-6-10) network were trained using standard Back-propagation and proposed modification. The networks were trained with different values of η and ß. For each set of η and ß, 5 trails were made. Simulation results showing the average training cycles required to reduce error to certain limit are summarized in Tables 5.5 and 5.6. ß was varied upto 3.0. A graphical representation of the results are shown in Figures 5.7 and 5.8. Results show that convergence is considerably faster when the modified Back-propagation is used with the value of ß within certain limit. Range of ß for which accelerated convergence can be achieved is also largely dependent on η. With larger value of η, ß should be kept lower. Larger values for both η and ß could result in oscillation in learning process and ultimately slow down the convergence rate. For example, Figure 5.8 shows that for η=0.6 and ß=2.5 convergence become slower when error limit was set to 0.001.

A typical convergence behavior for character recognition problem is shown in Figures 5.9 and 5.10. Figure 5.9 shows that even though certain value of ß (here ß=2.0 and η=0.8) can cause slight oscillation in the early stage of learning but adaptation of learning rate will eventually lead to stable convergence. On the other hand, as the Figure 5.10 shows, a suitable combination of η and ß yields a smooth learning resulting faster convergence. Fig. 5.11 illustrates how the effective learning rate η_{eff} adapts itself during oscillation in learning process. Eventually the learning rate and error decrement follow a smooth variation as the latter stage of learning.

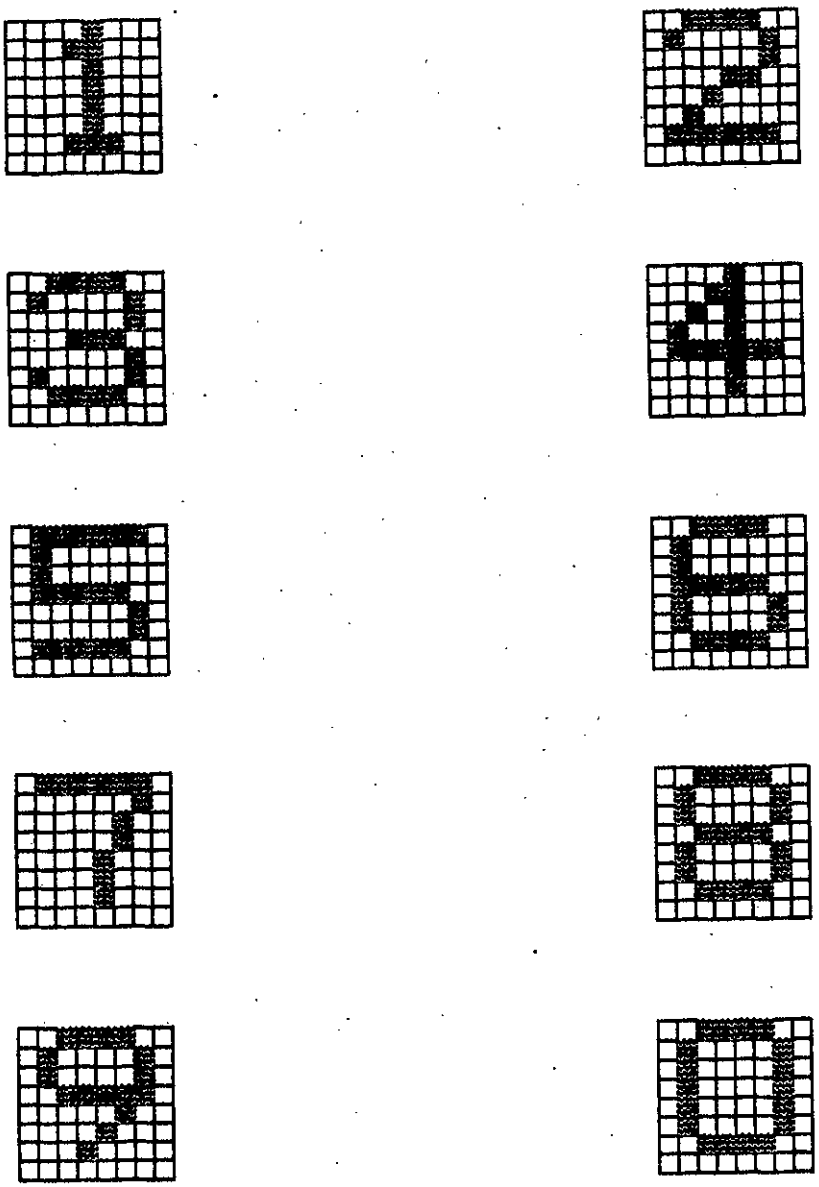


Figure 5.6 Ten alphanumeric digits (0∼9)

Table 5.5 Comparison of training cycles required to realize character recognition problem with a 64-5-10 network using standard Back-propagation and proposed modification. (a) For η=0.20 (b) For η=0.40 (c) For η=0.80

(a)

Error Limit	η=0.20							
	Standard BP	Modified B.P						
		β=1.0	β=1.50	β=1.75	β=2.0	β=2.25	β=2.75	β=3.0
0.01	1726	1592	1057	899	750	849	1237	1302
0.005	2866	2752	1832	1589	1317	1499	1709	2180
0.001	11297	11203	7517	6709	5504	6484	5756	8653

(b)

Error Limit	η=0.40							
	Standard BP	Modified B.P						
		β=1.0	β=1.25	β=1.50	β=1.75	β=2.0	β=2.25	β=2.50
0.01	832	792	642	512	441	457	1145	9594
0.005	1384	1367	1124	902	771	799	1502	9876
0.001	5471	5672	4675	3786	3184	3259	4154	12192

(c)

Error Limit	η=0.80							
	Standard BP	Modified B.P						
		β=0.5	β=0.8	β=1.0	β=1.25	β=1.35	β=1.50	β=1.75
0.01	412	817	490	399	316	287	325	426
0.005	688	1391	836	688	545	487	566	602
0.001	2741	5696	3392	2849	2210	2000	2394	2040

Table 5.6 Comparison of training cycles required to realize character recognition problem with a 64-6-10 network using standard Back-propagation and proposed modification. (a) For η=0.20 (b) For η=0.40 (c) For η=0.60

(a)

Error Limit	η=0.20							
	Standard BP	Modified B.P						
		β=0.75	β=1.0	β=1.25	β=1.5	β=2.0	β=2.5	β=3.0
0.01	1572	1953	1474	1144	827	585	592	883
0.005	2384	3047	2323	1842	1342	1030	964	1219
0.001	8546	11301	8011	6968	5006	4320	3670	3786

(b)

Error Limit	η=0.40							
	Standard BP	Modified B.P						
		β=0.75	β=1.0	β=1.25	β=1.5	β=1.75	β=2.0	β=2.5
0.01	732	913	622	468	372	343	296	772
0.005	1168	1491	1018	798	629	592	502	1015
0.001	4285	5538	3916	3250	2520	2431	1965	2915

(c)

Error Limit	η=0.60							
	Standard BP	Modified B.P						
		β=0.75	β=1.0	β=1.25	β=1.5	β=1.75	β=2.0	β=2.5
0.01	474	569	424	364	292	255	372	1085
0.005	742	923	686	605	477	409	527	1182
0.001	2788	3580	2676	2365	1815	1582	1683	2133

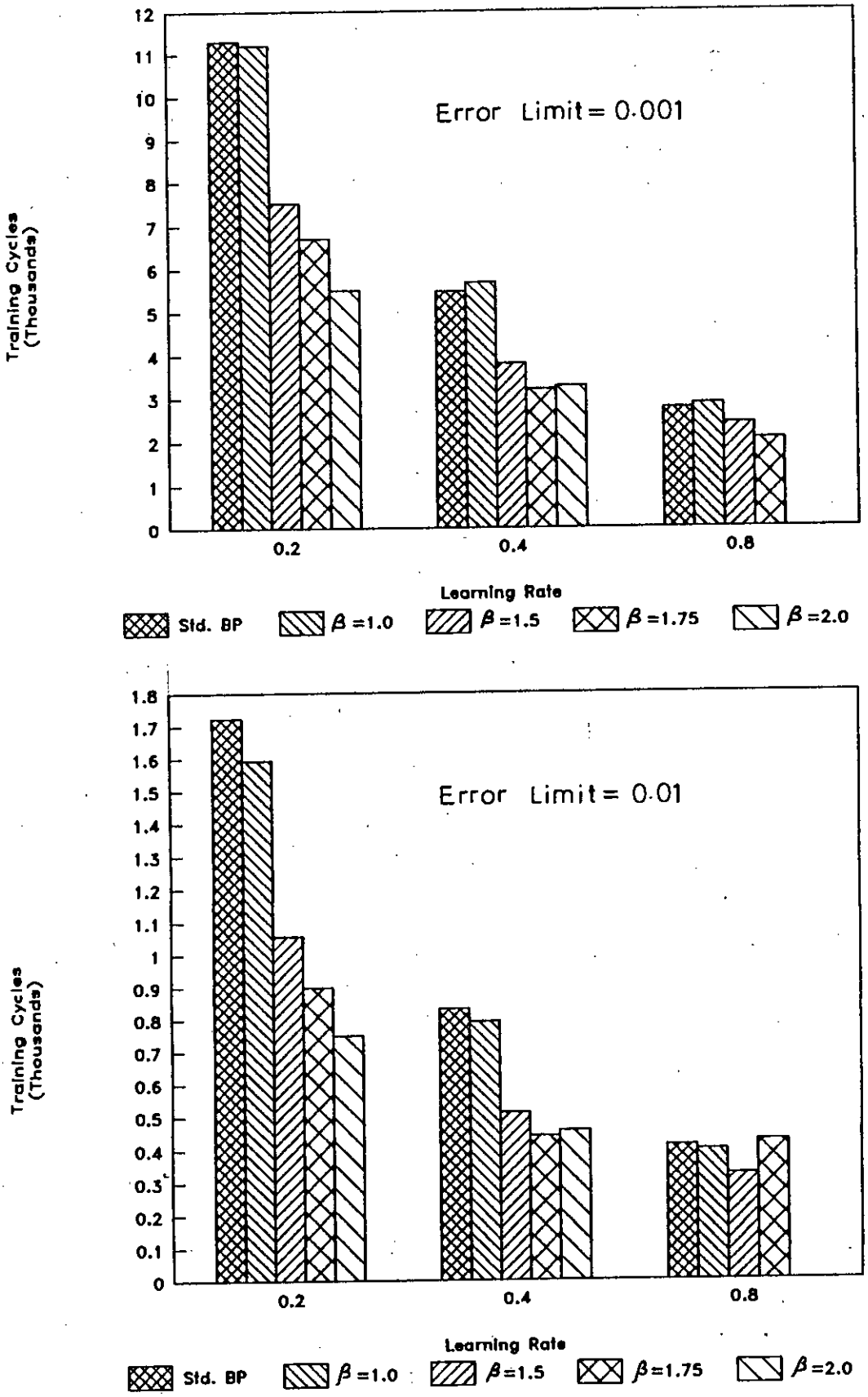
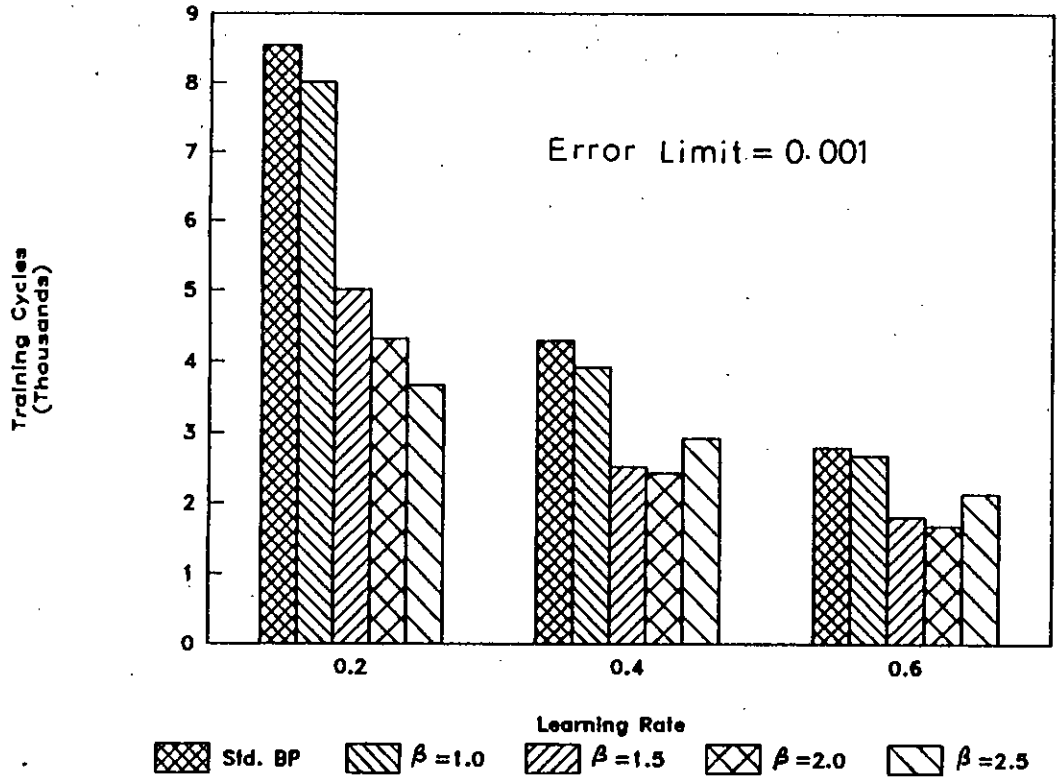


Figure 5.7 Comparison of convergence speed between standard Back-propagation and the proposed modification to learn character recognition problem with a 64-5-10 network for different values of η and ß.



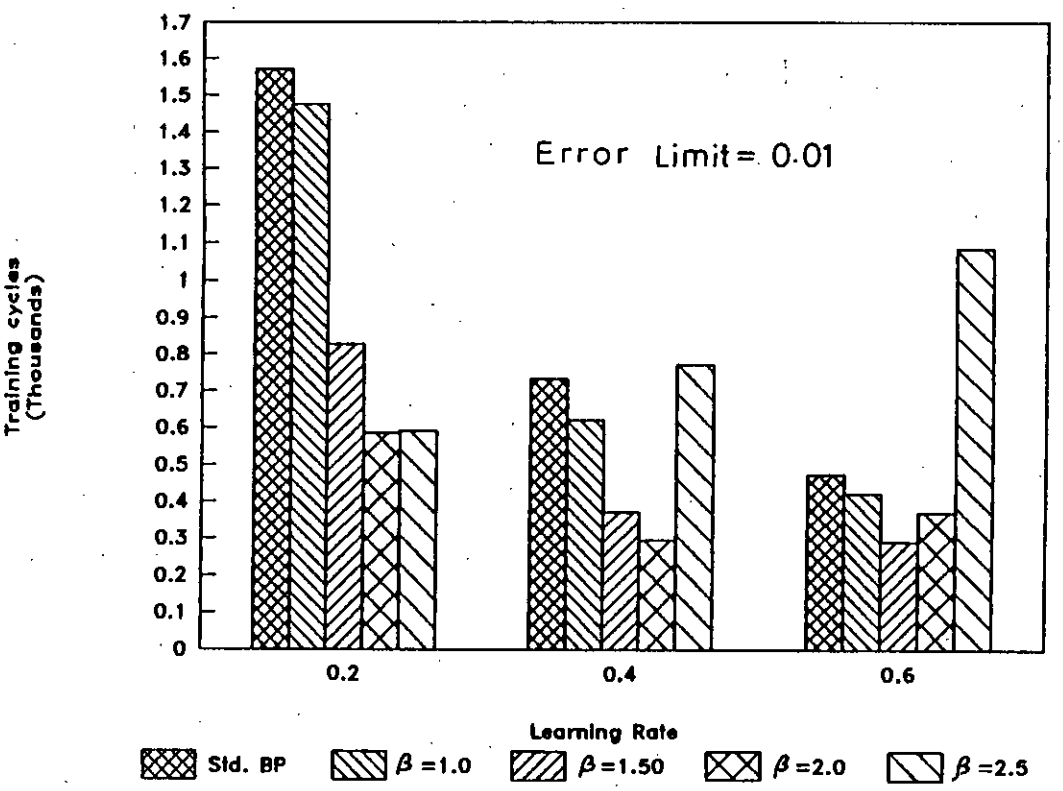


Figure 5.8 Comparison of convergence speed between standard Back-propagation and the proposed modification to learn character recognition problem with a 64-6-10 network for different values of η and ß.

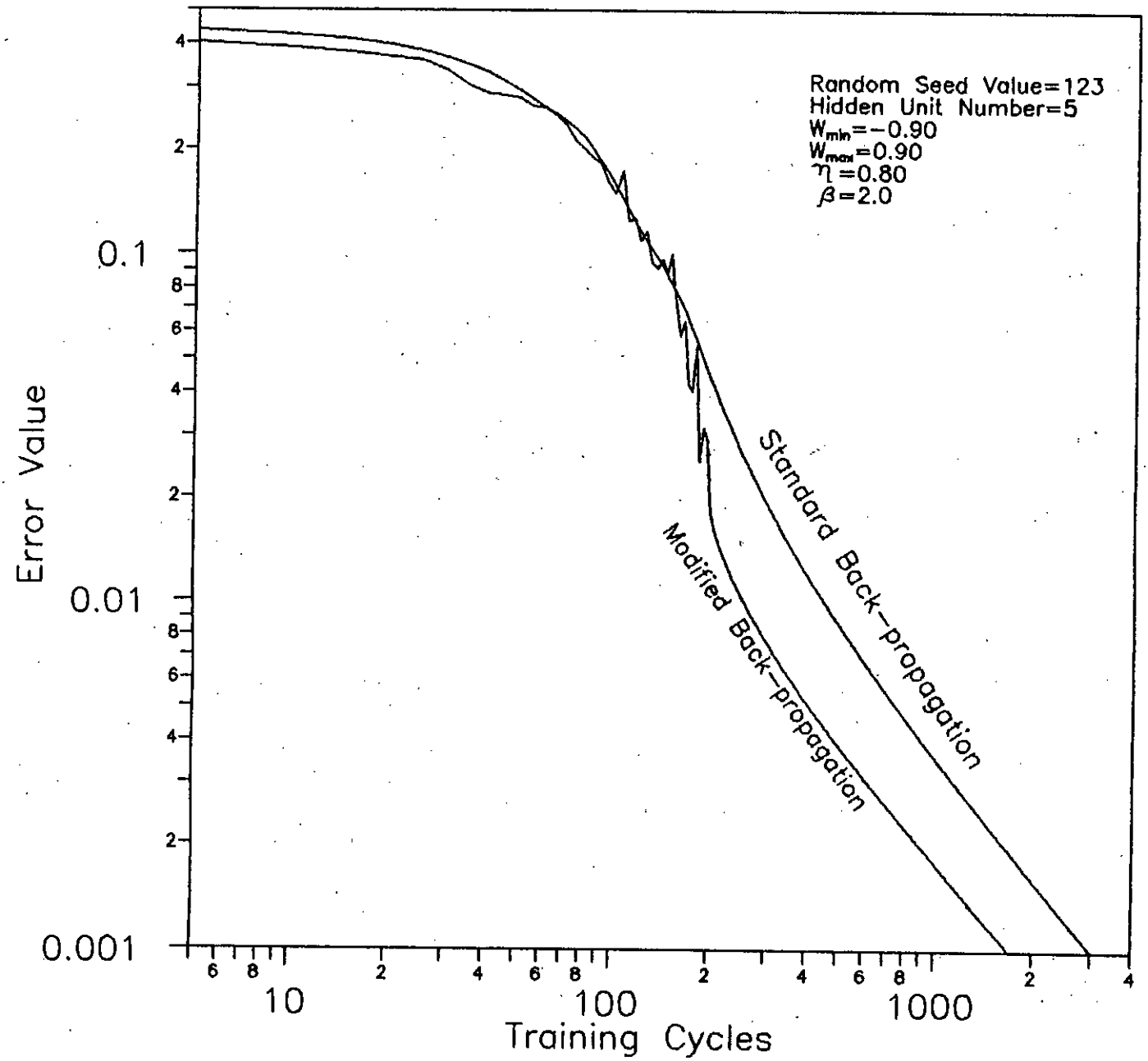


Figure 5.9 Typical learning characteristic for character recognition problem with 5 hidden units using standard Back-propagation and Proposed modification.

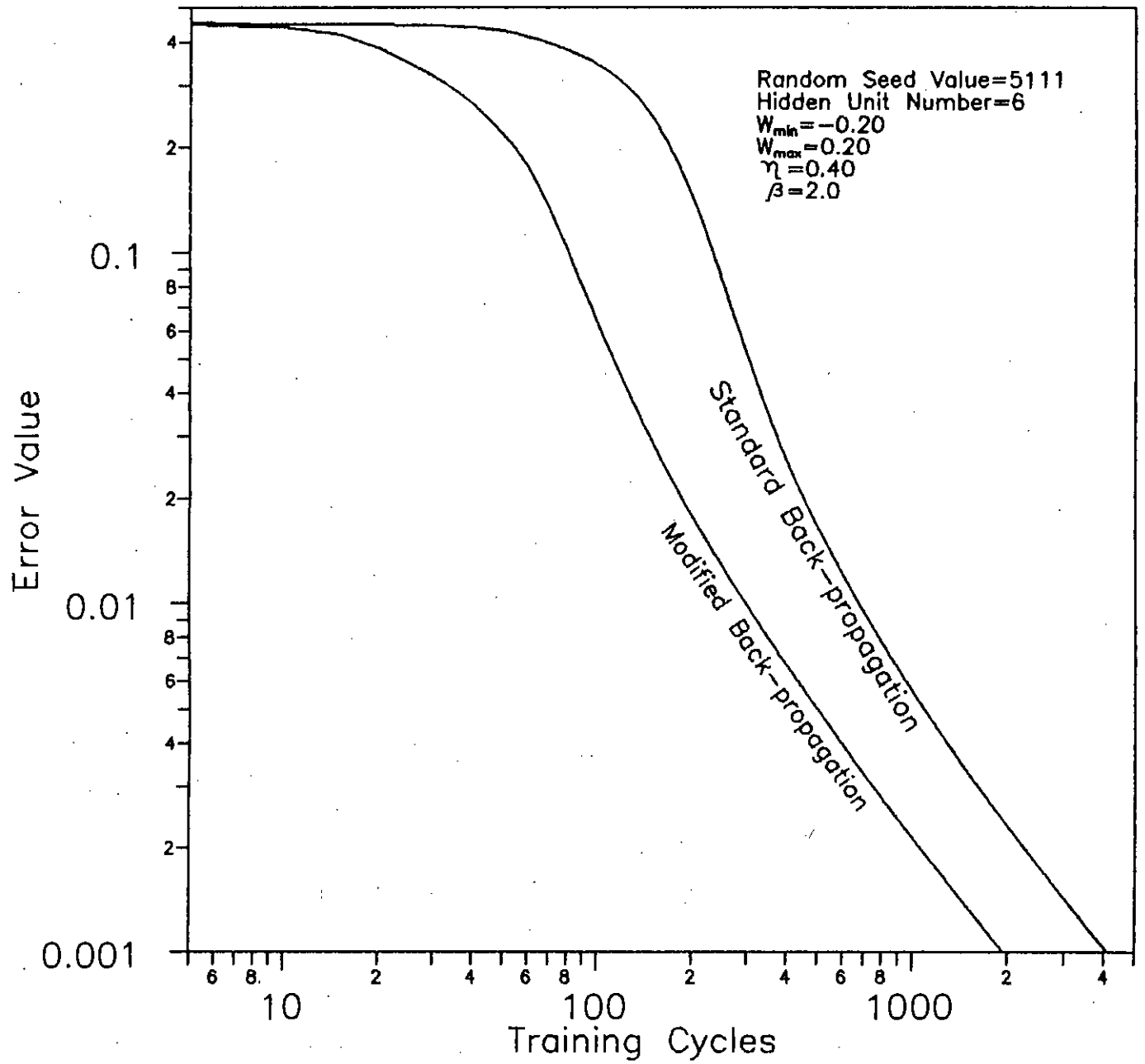


Figure 5.10 Typical learning characteristic for character recognition problem with 6 hidden units using standard Back-propagation and Proposed modification.

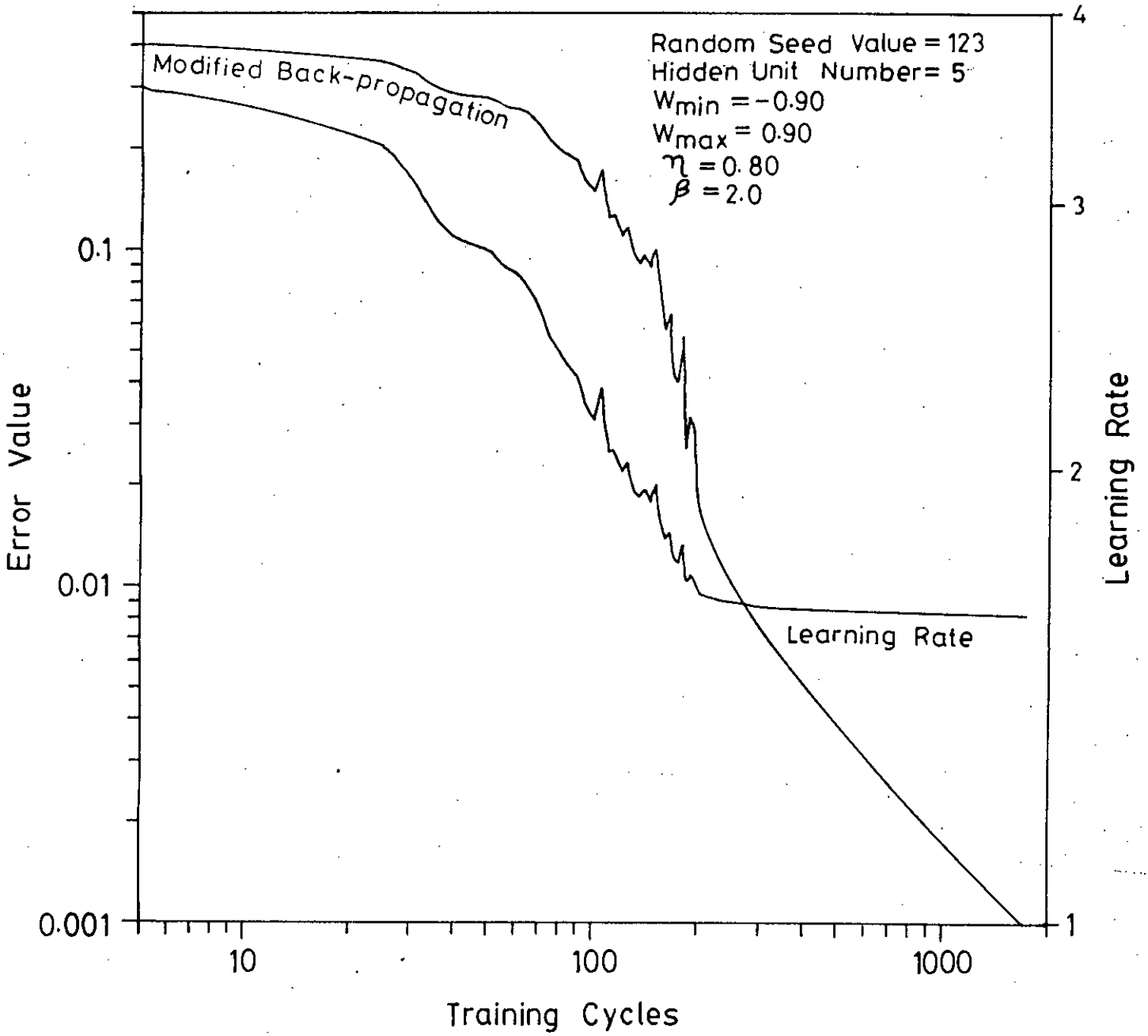


Figure 5.11 Variation of effective learning rate η_{eff} with error as learning proceeds in case of character recognition problem for a 64-5-10 network.

Table 5.7 Comparison of training cycles required to realize character recognition problem with a 64-5-10 network using standard Back-propagation with Log-likelihood (LL) cost function and modification as per Eq. (4-9). (a) For η=0.05 (b) For η=0.10 (c) For η=0.20

(a)

Error Limit	η=0.05						
	BP (LL)	BP modified as per Eq. (4-9)					
		β=0.50	β =1.0	β=1.50	β=1.75	β=2.0	β=2.5
0.01	824	1764	852	500	471	426	2195
0.001	2602	4965	2466	1421	1304	1252	3974

(b)

Error Limit	η=0.10						
	BP (LL)	BP modified as per Eq. (4-9)					
		β=0.25	β=0.50	β=0.75	β=1.0	β=1.25	β=1.5
0.01	509	1970	928	643	513	380	297
0.001	1241	5083	2429	1590	1230	1025	764

(c)

Error Limit	η=0.20						
	BP (LL)	BP modified as per Eq. (4-9)					
		β=0.25	β=0.50	β=0.75	β=1.0	β=1.25	β=1.50
0.01	226	915	419	282	197	153	400
0.001	584	2329	1086	749	559	447	673

As done in XOR, character recognition problem was also investigated using Log-likelihood cost function and the modification suggested in Eq.(4-9) Simulation results for different values of η and ß are summarized in the Table 5.7. Here also, for ß roughly within the range of $1.0 \sim 1.5$, the modification accelerate the convergence speed considerably.

5.4 *Function Approximation*

The first two problems studied earlier had binary input-outputs. Next the modification is intended to test with an analog input-outputs. The problem chosen is to approximate the sine function sin(x), for $-5.0<x<5.0$ radian. To do this the network needs single input and output. A 1-6-1 network and a 1-7-1 network were trained with values of sin(x) taken at the interval of 0.1 radian. Input and target values are the same. A total of 101 input-output values were trained by the network. An input sin(x) function and the output produced by the network in response to input is shown in Figure 5.12. With the sum-squared error of 0.0005, the network has approximated the sine function. With lesser error limit, the approximation will be better. In this case the activation function of output unit is linear instead of sigmoidal.

Simulation results showing the average of 5 trials are shown in Tables 5.8 and 5.9. Accelerated convergence is achieved in modified Back-propagation for values of ß roughly within the range of $0.25 \sim 1.15$. Typical convergence characteristics in this function approximation problem is shown in Figures 5.13 and 5.14. Figure 5.13 shows that with η=0.20, standard Back-propagation shows severe oscillation in learning whereas proposed

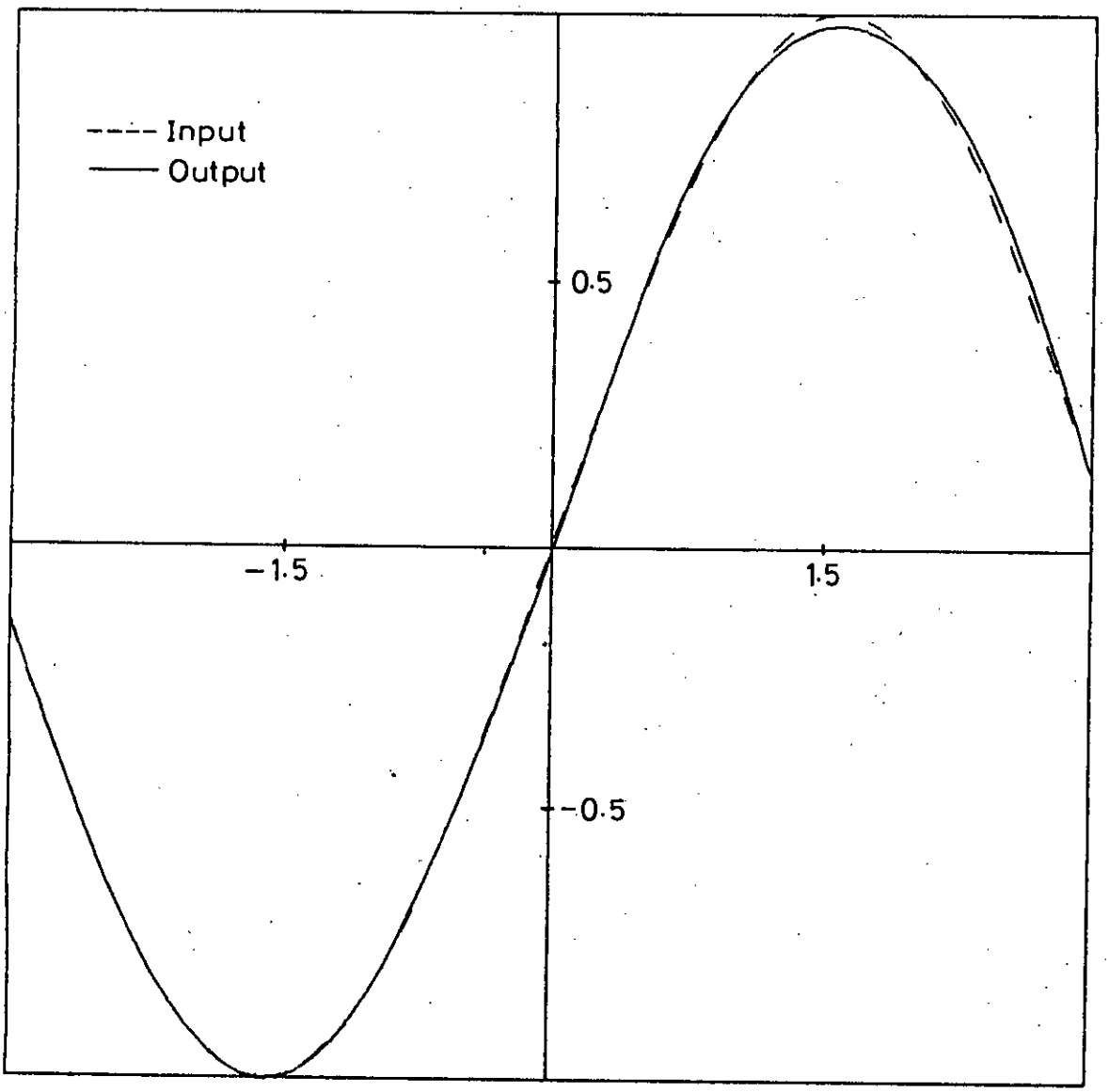


Figure 5.12 Input sin(x) function and output produced by the 1-6-1 network in response to input.

Table 5.8 Comparison of training cycles required to realize sine function approximation problem with a 1-6-1 network using standard Back-propagation and proposed modification. (a) For η=0.05 (b) For η = 0.10 (c) For η = 0.20

(a)

Error Limit	η =0.05						
	Standard BP	Modified B.P					
		β=0.5	β=0.75	β=1.0	β=1.05	β=1.10	β=1.15
0.001	2739	2863	2794	1810	1998	2768	2434
0.0005	8601	11361	9262	8261	7853	7390	7461

(b)

Error Limit	η=0.10						
	Standard BP	Modified BP					
		β=0.5	β=0.6	β=0.7	β=.0.8	β=0.9	β=1
0.001	1051	1124	1140	865	1521	1451	951
0.0005	6375	4806	4429	5750	5948	8772	7320

(c)

Error Limit	η= 0.20						
	Standard BP	Modified BP					
		β=0.1	β=0.25	β=0.35	β=0.45	β=0.5	β=.55
0.001	2776	6920	3541	3033	2171	2736	
0.0005	10940	14002	6804	6700	7352	6901	

Table 5.9 Comparison of training cycles required to realize sine function approximation problem with a 1-7-1 network using standard Backpropagation and proposed modification. (a) For η=0.075 (b) For η = 0.10 (c) For η = 0.15

(a)

Error Limit	η =0.075						
	Standard BP	Modified B.P					
		β=0.9	β=1.0	β=1.01	β=1.02	β=1.03	β=1.05
0.001	3033	2783	765	642	623	625	977
0.0005	7726	5916	4133	3960	3893	3470	3383

(b)

Error Limit	η=0.10						
	Standard BP	Modified BP					
		β=0.5	β=0.65	β=0.7	β=0.75	β=0.8	β=0.90
0.001	3610	4696	3896	3843	2410	2261	1295
0.0005	8320	7253	5916	5773	3850	6150	5556

(c)

Error Limit	η= 0.15						
	Standard BP	Modified BP					
		β=0.30	β=0.35	β=0.40	β=0.45	β=0.50	β=0.55
0.001	2613	4976	4756	3456	2793	2533	2853
0.0005	8823	7470	7433	5513	5120	8166	6356

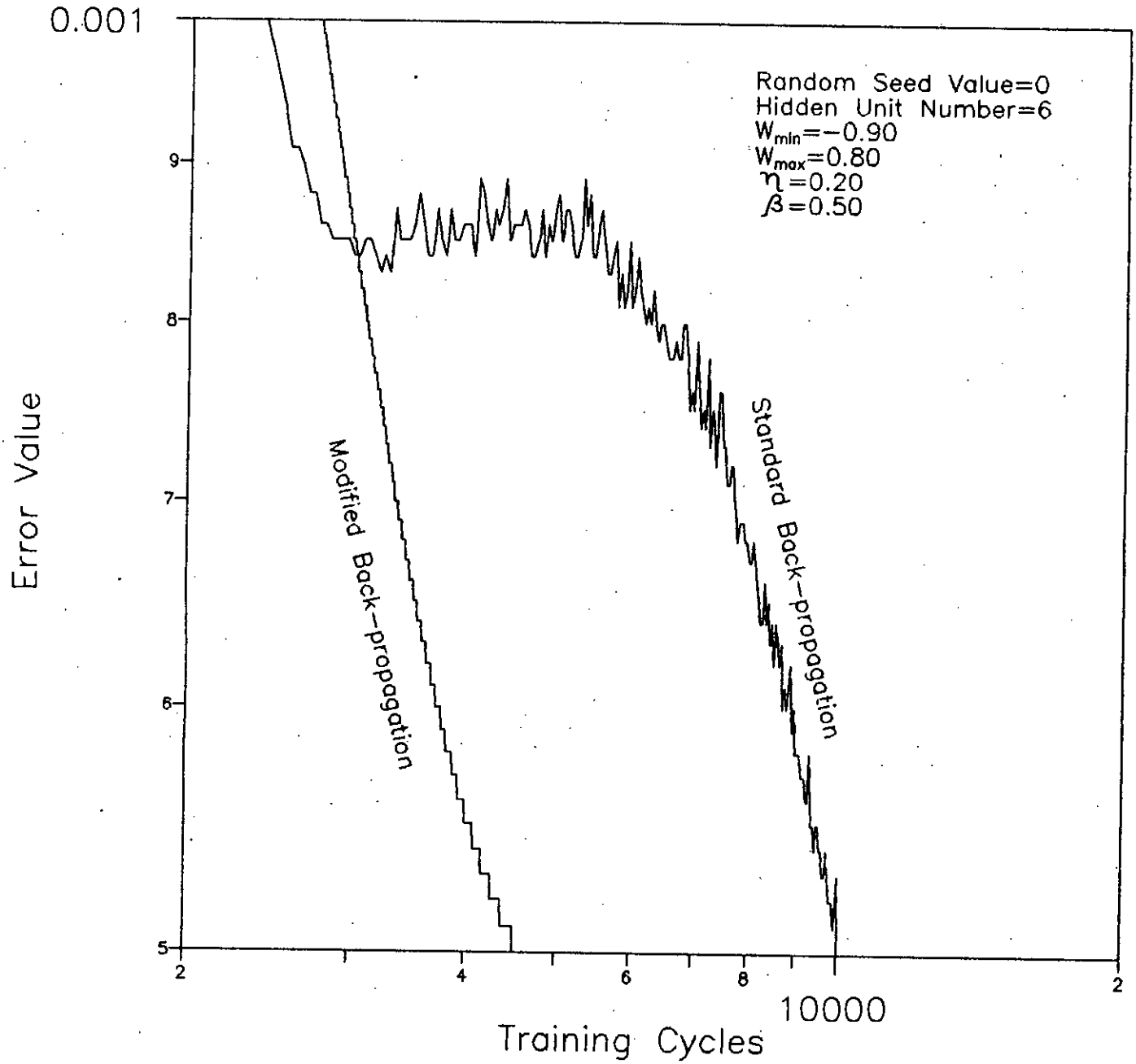


Figure 5.13 Typical learning characteristic for learning sine function with 6 hidden units using standard Back-propagation and Proposed modification.

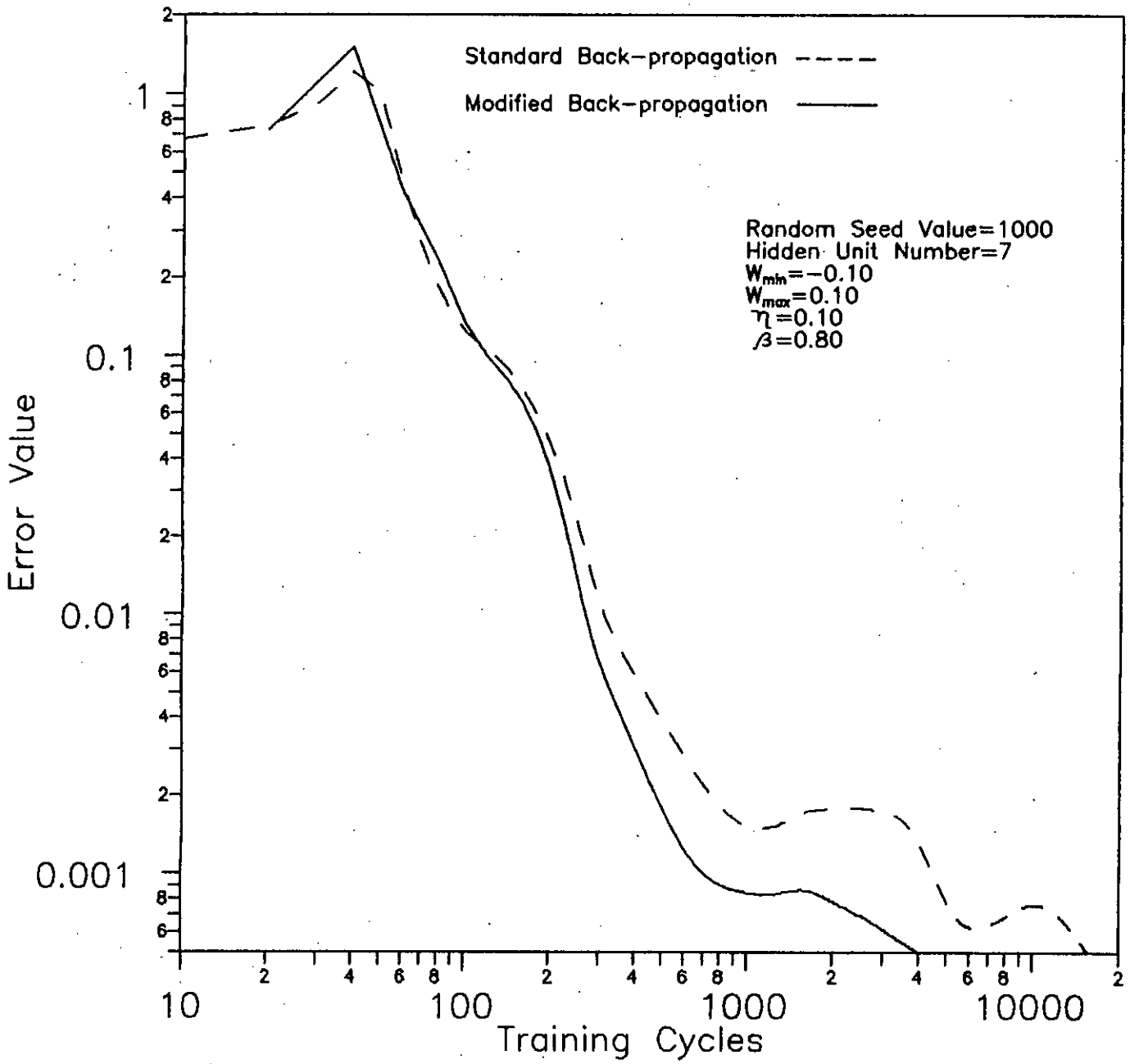


Figure 5.14 Typical learning characteristic for learning sine function with 7 hidden units using standard Back-propagation and Proposed modification.

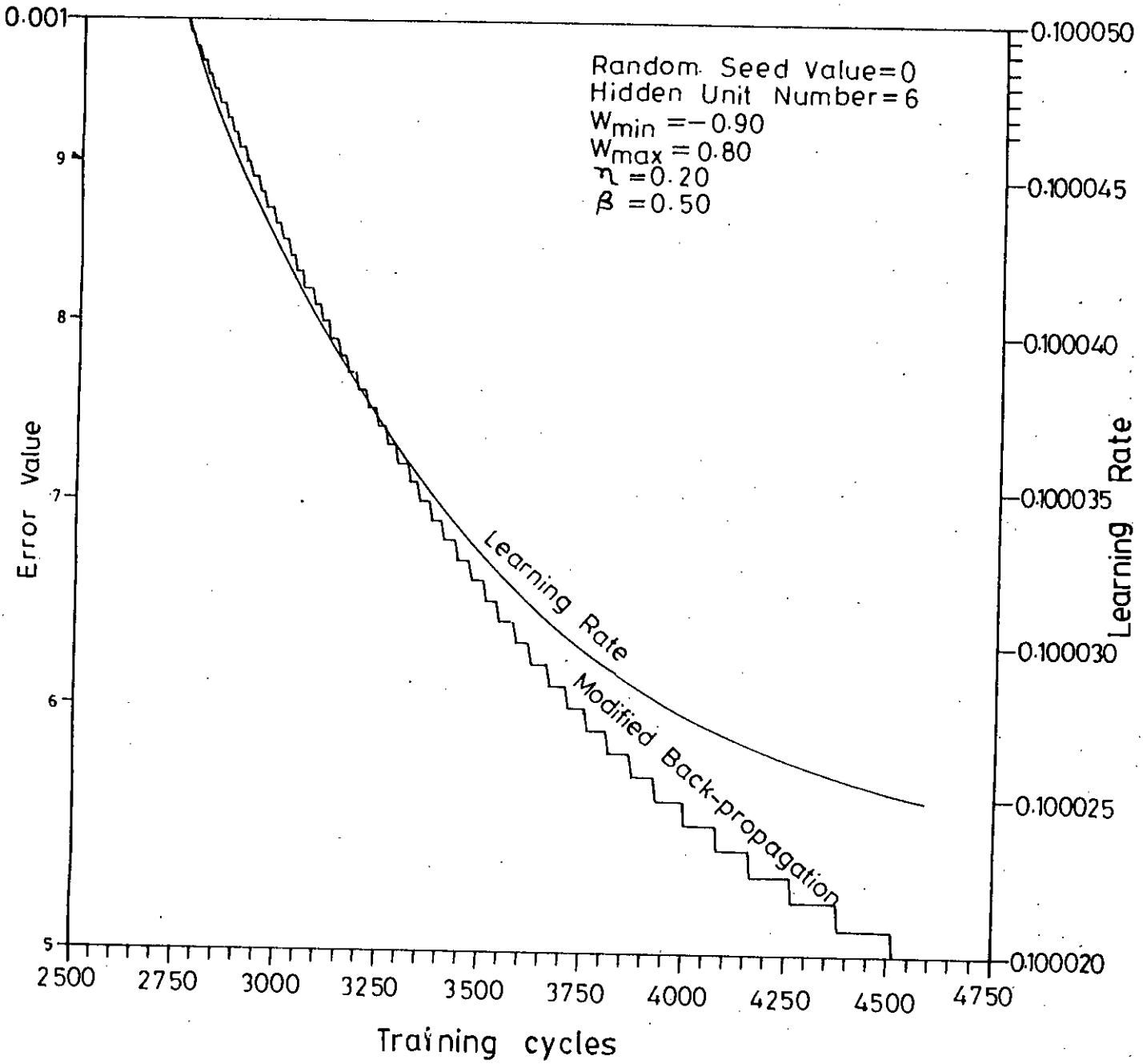


Figure 5.15 Variation of effective learning rate η_{eff} with error as learning proceeds in case of sine function approximation for a 1–6–1 network.

modification yields smooth gradual decrement of error. Modification takes only half of the number of training cycles required by standard Back-propagation. Since suitable value of η for good convergence is problem specific the value of η=0.20 appears to be large for standard Back-propagation and therefore causes oscillation. But with the same value of η, the effective learning rate is adaptively reduced to suitable value eliminating oscillation and making learning faster. This is well illustrated in Figure 5.15.

5.5 *Summary*

In this chapter, simulation results comparing the convergence speed of standard Back-propagation algorithm and the modification proposed in the previous chapter are presented. Simulation has been carried out with three different problems. Two problem have binary input-outputs and the other has analog input-output. Results show that, for certain value of η, choice of suitable value of ß will greatly accelerate the convergence speed. However, like η, suitable value of ß for accelerated convergence is problem specific. It has also been observed that for larger value of η, the range of ß for which accelerated convergence can be achieved becomes reduced. For larger values of η, ß should be kept lower. From the simulation results, the value of ß for binary input-output can be roughly recommended within 1.0~2.0 and that for analog input-output within $0.25 \sim 1.15$.

In designing a neural network based system, the usual practice is to train many networks and select the one which yields the best performance in terms of generalization ability and/or desired response. This needs training networks with different initial weights and other parameters. In such cases, once a suitable value of ß can be found, training of different candidate networks with ß around that value will save considerable amount of computational time.

CHAPTER 6

CONCLUSION

CONCLUSION

FUTURE WORK

6.1 *Conclusion*

The present work was aimed to accelerate the convergence speed of Back-propagation learning which is the most widely used learning algorithm in neural network. Standard Back-propagation uses sum of squared error measured at output layer as the cost function and updates weights to minimize this error in steepest descent method. During training the network, one parameter η called learning rate which determines the step size in weight change is kept constant. In the present work, a new cost function expressed as an exponential function of the sum of squared error measured at the output layer is defined. Use of this cost function makes the effective learning rate parameter vary with error during training. However, this requires one additional parameter ß to be selected. Simulation results with two problems which have binary input-output relationship and one problem which has analog input-output relationship show significant improvement in convergence speed within certain suitable range of ß. Like η in standard Back-propagation, selection of suitable range of ß which yields accelerated convergence is also problem specific. Higher values for both η and ß will cause oscillation in the learning process. If a higher value of η is selected, ß should be kept lower. It has been observed in the simulation that, in cases where certain value of η which cause severe oscillation in learning by standard Back-propagation, yields smooth learning in the proposed modification with a suitable value of ß. This is because the modification adapts effective learning rate during training. Thus suitable choice of ß in this modification is as much crucial as choice of η in the standard Back-propagation.

Through the observation of the simulation results a range of ß is recommended in the present study. This recommended range is $1.0 \sim 2.0$ for binary input-output relationship and $0.25 \sim 1.15$ for analog input-output relationship. This recommended value does not guarantee accelerated

convergence by the proposed modification for all problems but gives an initial starting point to search for a suitable value of ß. In designing a neural network based system usually a set of candidate networks are trained. Each network ultimately settles down to a different weight set. The performance of each network in terms of generalization ability or desired response is tested, and the best one is selected. Thus, in situations where one has to train many candidate networks, once a suitable value of ß is found, training networks with ß around that value can save considerable computational time.

6.2 *Future Work*

In the present work, convergence speed of the proposed modification is compared with that of standard Back-propagation and the modification proposed by Halt J. [10]. There are other modifications proposed by several researchers [5]-[8], [11]-[12] and the present modification has not been compared with those. It is necessary to compare with other modifications in order to have a complete comparison. One of the most important features of neural network is its generalization ability. If a learning algorithm converges very fast but does not possess good generalization ability, that algorithm is less attractive in almost all applications. The present study dealt only with convergence speed of the proposed modification. In order to evaluate the modification completely, its generalization ability needs to be investigated and compared with that of standard Back-propagation.

REFERENCES

REFERENCES:

[1] Ronald J. MacGregor. *Neural and Brain Modeling.* Academic Press, San Diego, CA,1987.

[2] Robert A. Lavine. *Neurophysiology: The Fundamentals.* The Collamore Press, Lexington, MA,1983

[3] Donald O. Hebb. *The Organization of Behavior.* Wiley, New York, 1949.

[4] D.E. Rumelhart, J.L. MacClelland and the PDP research Group, *Parallel Distributed processing,* Vol. 1, M.I.T. Press, 1986

[5] T.P. Vogl et. al., "Accelerating the convergence of the backpropagation method", *Biol. Cybern.,* Vol. 59, pp. 257-263, 1988.

[6] L.W. Chan et. al., "An adaptive training algorithm for backpropagation networks", *Computer Speech & Language,* Vol.2, pp.205-218, 1987.

[7] R.S. Sutton, "Two problems with backpropagation and other steepest-descent learning procedure for networks", *in Proc. 8th Annu. conf. of the Cognitiv Science Society,* pp. 823-831, 1986.

[8] R. Jacobs, "Increased rate of convergence through learning rate adaptation", *Neural Networks,* Vol. 1, No. 4, pp. 295-307, 1988.

[9] L.M. Reyneri et. al., "An analysis on the performance of silicon implementation of backpropagation algorithms for artificial neural networks", *IEEE Trans. Comput.,* Vol. 40, No. 12, pp. 1380-1389, 1991.

[10] Holt. M.J.J. and Semnani, S. "Conyerge of backpropagation in Neural network using a log-likelihood cost function", *Electronics Letters*, Vol. 26, No. 23, pp. 1964-1965, 1990.

[11] Samad Tariq, "Back Propagation With Expected Source Values", *Neural Networks*, Vol. 4, pp. 615-618, 1991.

[12] S.C Huang and Y.F. Huang, " Backpropagation with selective updates", *in Proc. IEEE int. Conf.* Neural Networks, San Diego, CA 1988, pp. 584-589.

APPENDIX

Appendix: Flowchart of Backpropagation Algorithm