1518

NON-STATIC LEARNING FOR NEURAL NETWORKS

Dissertation submitted to the Jawaharlal Nehru University in partial fulfilment of the requirements for the award of the Degree of

MASTER OF TECHNOLOGY

in

COMPUTER SCIENCE & TECHNOLOGY

66p + fig.

by

DWARIKA NATH MISHRA

SCHOOL OF COMPUTER AND SYSTEMS SCIENCES
JAWAHARLAL NEHRU UNIVERSITY
NEW DELHI - 110 067
INDIA
JANUARY 1993

CERTIFICATE

This is to certify that the thesis entitled **NON-STATIC LEARNING FOR NEURAL NETWORKS**, being submitted by me to Jawaharlal Nehru University in partial fulfilment of the requirements for the award of the degree of **Master of Technology**, is a record of original work done by me under the supervision of Prof. K.K. Bhardwaj, **School of Computer Systems & Sciences**, J.N.U., during the Monsoon Semester, 1992.

The results reported in this thesis have not been submitted in part or full to any other University or Institution for the award of any degree etc.

[illegible]

(DWARIKA NATH MISHRA)

[illegible]

Dr.R.G.Gupta
Professor & Dean 5/1/93
SC & SS, JNU
New Delhi 110 067

[illegible] Jan 5, 1993

Dr.K.K.Bharadwaj
Professor,
SC & SS, JNU
New Delhi 110 067

ACKNOWLEDGEMENTS

I wish to express my sincere and heartfelt gratitude to Prof. K. K. Bharadwaj, School of Computer and Systems Sciences, Jawaharlal Nehru University, for the unfailing support he has provided throughout. In all respects, I am very grateful for the patience he has exibited and for the time he has spent with me discussing the problem. It would have been impossible for me to come out sucessfully without his constant guidance.

I extend my thanks to Prof. R.G. Gupta, Dean, School of Computer and Systems Sciences, JNU for providing me the oppertunity to undertake this project. I would also like to thank the authorities of the school for providing the necessary facilities to complete my project.

I am also grateful to my colleagues for their encouragements, suggestions and worthy discussions. I acknowledge and thank each and everyone of those who directly or indirectly helped me in this work.

(Dwarika Nath Mishra)

Dedicated to
my parents

ABSTRACT

The size of the neural network depends on the complexity of the learning task; more hidden units are needed to approximate a complex function. Higher structured learning is defined as not only modifying the strength of the connections but also the overall topology of the network to get the optimal size for a particualr class of problems. An outline of the segretion algorithm which avoids the principal inadequacies of static algorithms is presented. A non-static learning paradigm based on segregation algorithm that allows creation of units and their interconnections during learning is proposed and implemented. Also the relationship between the network structure and the ability of the network to generalise from the training sets has been examined and computational results presented.

CONTENTS

CHAPTER 1

INTRODUCTION

The quest for efficient computational approaches to artificial intelligence has undergone a significant evolution in last few years. Leading laboratories around the world are pushing towards brainlike computers-the sixth generation. Engineered intelligent systems behave with remarkably rigidity when compared with their biological counterparts,to recognise objects or speech,to manipulate and adapt in an unconstructed environment and to learn from the past experience. They lack commonsense knowledge and reasoning, knowledge structures for recognising complex patterns and fails to point out their own limitations. A major reason for this limited technical success in emulating one of the fundamental aspects of human intelligence lies in the difference between the organisation and structuring of knowledge and dynamics of biological neuronal circuitry and its simulation using the symbolic processing paradigm [14].

Rapid advances in computational and behavioral theories have brought about a new and much more sophisticated effort to model cognition and perception in physiological plausible terms. In order to build intelligent machines, one obvious idea for AI researchers is to simulate the functioning of the brain directly on the computer. Over the past three decades, AI researchers have been doing extensive research in the areas of pattern recognition, adaptive machine learning, perception and sensory motor control [1,11]. Subsequent development of intelligent systems has pursued two different schools of thought ; one symbolic and the other is neurobiological, subsymbolic or connectionist. The past few years has witnessed a significant reawakening of interest in massively parallel computation, often portrayed as "neural networks" - a biologically inspired, computational and information paradigm [14].

1.1 NEURAL NETWORKS - AN AI TOOL

Neural networks represent a means i.e. a tool for understanding aspects of artificial intelligence. They can be treated as a way of representing knowledge. In a different perspective, neural networks are massively parallel computational paradigm that offers a new way of viewing problems in AI [4]. Because of their different structure neural networks can benifit considerably from customized architechtures. Artificial neural networks constitute an enhancing not a replacing technique of AI.

In neural networks we usualy start with a computational structure which can be mathematically formulated and studied. Neural networks employ distributed parallel processing to perform computation. The storage, processing and communication of information in neural networks occurs throughout the whole network rather at specific sites or memory locations. Thus memory and processing in neural networks are global rather then being local in nature. Computation by neural networks emerges spontaneously from fundamental physical principles. A neural network can be considered as a relaxation system which settles into a solution. Neural networks can compute significant results in a small number of steps after learning which is a fundamental and essential aspect of neural networks. The most striking and impressive characterstic of neural networks is the ease and naturalness with which they can learn. Neural networks are trained on a finite set of training example after which they become able to extrapolate on them in order to provide outputs never encountered before. They also remain quite robost in noisy environments. The ability of current VLSI technology to provide large number of simple processing elements(both analog and digital) allows for a quantum improvement in the cost performance of neural networks [4].

1.2 HISTORICAL PERSPECTIVE

For understanding the present state of development of neural networks it is essential to have at least an inlinking of the history of their development [10,14]. This history is enormously fascinating .

1.2.2 The early years

The science of artificial neural network made its first significant apperance in 1940s. Researchers desiring to duplicate the funtions of human brain developed simple hardware (and later software) models of biological neurons and its interconnection systems.In 1943, McCulloch and Pitts published the first systematic study of artificial neural network where they had shown that network of neurons with binary response functions was capable of computation. In 1949, Donald Hebb proposed a model for learning in neural networks through strengthening the connnections between elements connecting the input stimulus neurons and output patteren neurons . Later, Dean Edmonds and Marvin Minsky built an electromechanical learning machine which incorporated these ideas.

In the late 1950s Frank Rosenblatt of Cornell University invented a type of neural network which he called the *perceptron*. Minsky and Papert proved mathematically the capabilities and limitations of single layered ,linear perceptron of the type promoted by Rosenblatt.They found that though they worked quite well on very simple problems but their performance detoriated very rapidly as the tasks assigned to them got harder.

1.2.2 The dark ages

The failure of perceptrons dampened the enthusiasm for research in this type of device and brought research on neural networks to a virtual halt during 1970s. While many researchers abandoned neural networks to reorient their work along traditional symbol processing approaches,a few workers pursued the path suggested by the early perceptron work.

Dr. Benard Widrow first built *Adeline,* an adaptive linear neuron computer [11]. Adeline used negative feedback to overcome some of the problems of the perceptron.This work was extended to multiple neuron Madeline which demonstrated the capability of recognising spoken words and visual patterns independent of translation. On the theoretical side, Stephen Grossberg publised extensive mathematical analysis which establised the foundation for asociative learning in neural networks.

1.2.3 Rebirth of connectionism

In 1980s ,several events occured which reestablished neural network research as a credible endeavor. Amoung the most significant events which revived the study of neural networks was the publication of John Hopfield on *neural networks and physical systems with emergent collective computational abilities* in 1982. He presented simple analog circuit of neural network and used it to solve practical problems. Later the PDP(Parallel Distributed Processing) study group was established and a lot of research works were published.

1.3 THE NEURON MODEL

Artificial neural networks are patterned after real neural networks [10,14]. The basic computatinal element of real neural networks is a neuron which is shown in figure 1.1 . The four basic parts of a real neuron such as those found in the brain are (1) the synaptic buttons or synapses which serves as output devices, (2) the cell body which sums the membrane potentials provided by the synapses and fires at a rate which is a nonlinear function of the total voltages, (3)the axons which carries the electrical signals from the cell boody to subsequent synapses, (4) the dendrites ,branchlike structures which provide sensory input to the cell body. The cell body sums the membrane potentials between the synaptic buttons and fires voltage spikes down its output axon at a rate which depends on the sum of the input voltages. This dependence has been found to be sigmod in shape. It has been found experimentally that synaptic connections can produce eithor excitory or inhibitory effects. Excitory neurons greatly outnumber inhibitory neurons and tend to respond more rapidly to changes to input. The model of a neural network is illustrated in fig 1.3 . Most general neural network models assume a complete interconnection between all neurons and resolve the cases of nonconnected neurons (i,j) by setting the connection strength $T_{i,j} = 0$. The output of the neuron n is generated by multiplying the sum of the input voltages weighted by synaptic weights or connection strengths $T_{i.j}$ between neuron i and neuron j by nonlinear transfer function. There are however number of system design parameters [10] which must be specified for any neural network model such as :

* The structure of the system i.e. the number of layers
* The synchrony of the system

* Symmetry of the interconnections
* Feedback structure employed
* The transfer or activation function
* The formulation of learning strategy

1.4 ARTIFICIAL NEURAL NETWORKS TODAY

There have been many impressive demonstrations of artificial neural network capabilities [11]. A neural network has been trained to convert text to phonetic representations which were then converted to speech by other means(Sejnowsky and Rosenburg 1987). Some other networks have been developed which can recognise handwritten characters. A neural network based image compression system has been devised. Many of them use backpropagation algorithm discussed later . Fig.1.3 shows the block diagram of an advanced neural computing system. Real world signals are converted into discrete form at the interface block. The neural signal processing system handles the converted signal and the outputs are transferred to digital computer for further manipulations.

Neural network products are also commercially available today [10]. In the first International Conference on Neural Networks at San Diego, at least six commercial neural network systems were demonstrated. The Hecht-Nielson Neurocomputer Corporation has been established by Dr. Robert Hecht-Nielson which markets a neural network system called ANZA. The Nester Inc.has developed a product which recognises handwritten characters on checks and documents.

1.5 APPLICATIONS

Artificial neural networks have been proposed for tasks ranging from battle field management to minding the baby. Potential applications are those where human intelligence functions effortlessly and conventional computation has been proven cumbersome and inadequate. The conventional computers are extremely good at numerical computations and executing sequences of instructions that has been precisely formulated for them. On the other hand, neural networks prove themselves exellent in the field of perception. Successful neural network applications have several common characteristics [5] incuding :

* Applications are data intensive and dependent on multiple interacting parameters.

* Problem area is rich in historical data or examples

* Data set is incomplete, contains errors and describes specific examples

* Discriminator or function to determine solutions is unknown or expensive to discover

Several classes of applications are amenable to a neural network approach; most involve eithor pattern recognition or statistical mapping. Common applications include character recognition, forecasting, information processing, process monitoring, signal processing and robot control.Conversely, other applications do not lend themselves to a neural network approach. For example,mathematically accurate and precise applications, such as accounts receivable and inventry are unlikely candidates. In addition, applications that require temporary data load,analysis and reporting including sales data analysis and resource management are unsuitable. In general, applications that require deduction and stepwise logic are also not good choices for neural networks.

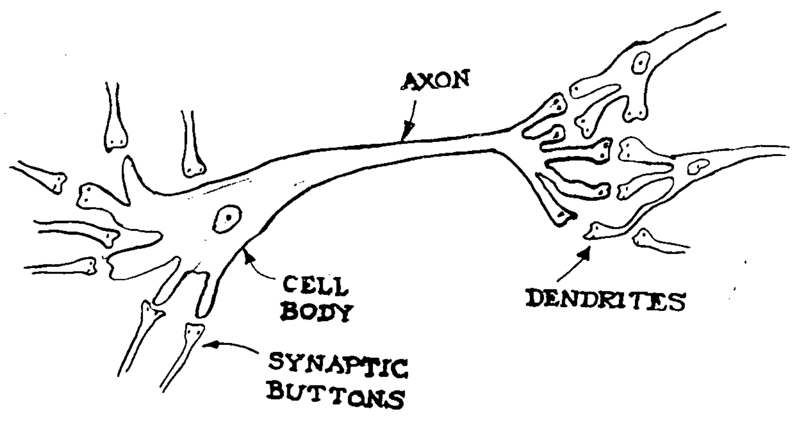


FIG.1.1 A TYPICAL BIOLOGICAL MODEL

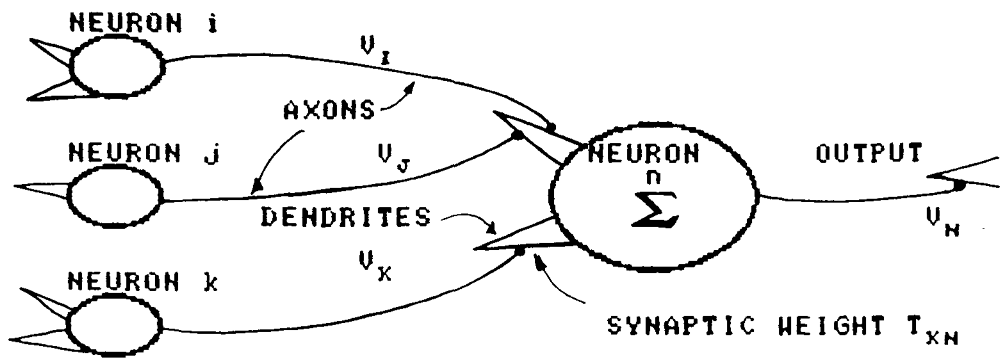


FIG.1.2 MODEL OF NEURAL NETWORK

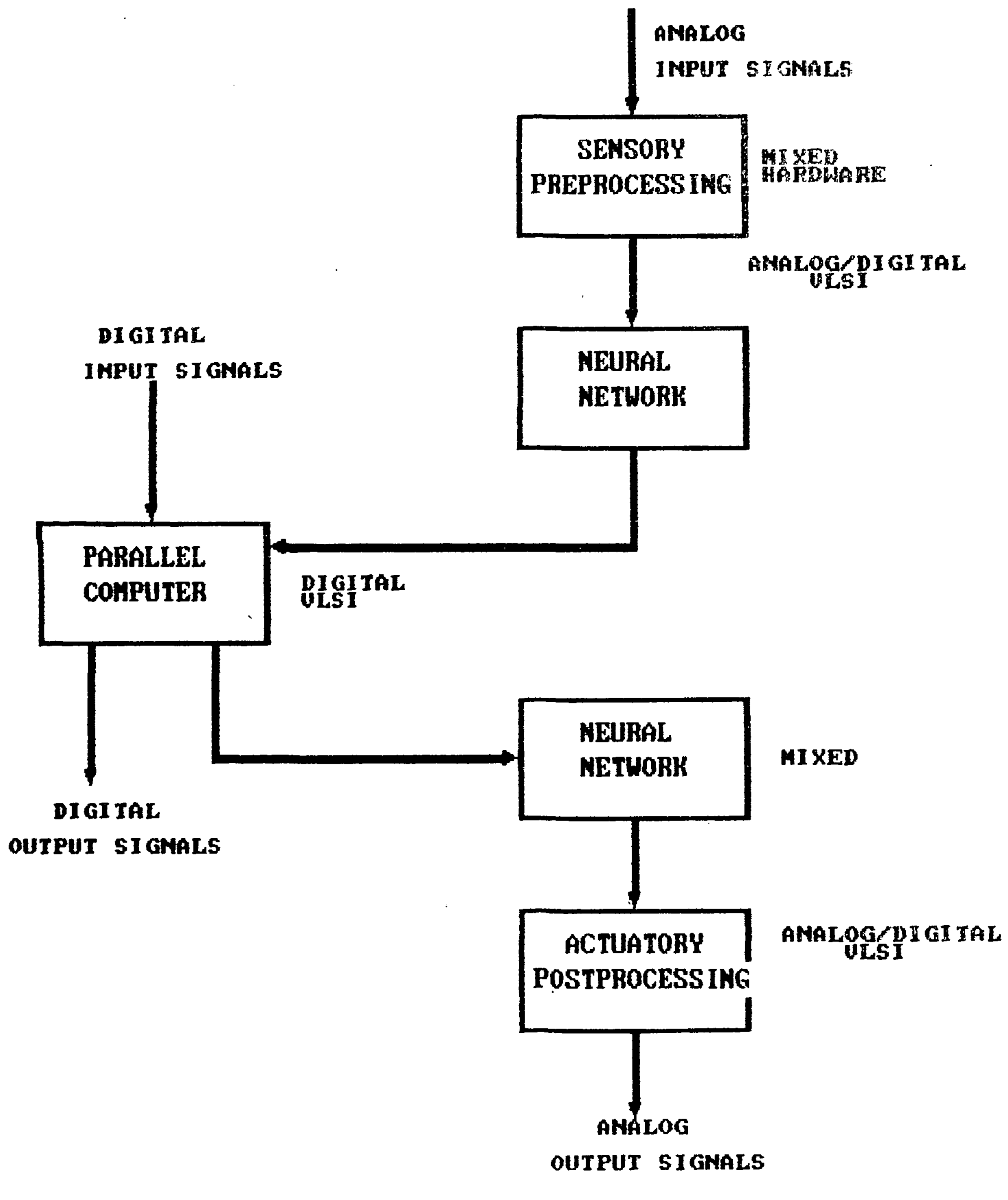


FIG.1.3 ADVANCED NEURAL COMPUTING SYSTEM

CHAPTER 2

STATIC AND NON-STATIC NEURAL NETWORKS

The human brain performs the formidable task of sorting a continuous flood of sensory information received from the environment and use them in an organised way. As already stated artificial neural networks always lags behind the human brain as far as dynamics and parallelism is concerned. In the real world, a neural network will be exposed to a constantly changing and noisy environments. So structing and organising a neural network is an emerging problem today.

2.1 GENERAL ARCHITECHTURE

In neurocomputing , the word *arhitechture* is reserved for the formal mathematical description of a neural network [13]. So the definition of neural network architechture has nothing to do with that architechture's implementation(i.e. the way in which neural network is implemented in software, neurosoftware and/or hardware). A neural network can be described more precisely as a parallel distributed information processing structure in the form of a directed graph , with the following sub-definitions and restrictions [13] :

1 . The nodes of the graph are called *processing elements*.

2 . The links of the graph is called *connections*.Each connection functions as an instantaneous unidirectional signal-conduction path.

3 . Each processing element can receive any number of incoming connections (also called input connections).

4 . Each processing element can have any number of outgoing connections , but the signal in all of these must be the same. In effect, each processing elemen that has a single output connection that can branch or fan out into copies to form multiple output connections(sometimes called collaterals),each of which carries the same identical signal.

5 . Processing elements can have local memory.

6 . Each processing element posseses a transfer function which can use (and alter) local memory,can use input signals and which produces the processing element's output signal. In other words, the only inputs allowed to the transfer function are the values stored in the processing element's local memory and the current values of the input signals in the connections received by the processing element. The only outputs allowed from the transfer function are values to be stored in the processing element's local memory and the processing element's output signal.Transfer functions can operate continuously or episodically.If they occur episodically , there must be an input called *activate* that causes the processing element's transfer function to operate on the current input signals and local memory values and to produce an updated output signal. Continuous processing elements are always operating. The activate input arrives via a connection from scheduling processing element

that is the part of the network.

7 . Input signals to a neural network from outside arrive via connections that originate outside.Outputs from the network to outside are connection that leave the network.

Internal details of a processing element in neural networks is shown in Fig 2.1 . In addition to the structure presented above, almost all known neural networks have their processing elements divided into disjoint subsets called *layers* or *slabs* in whuch all of the processing elements posses essentially the same transfer function.

Processing element transfer functions usually have a subfunction called a *learning law* that is responsible for adapting the input-output behavior of the processing element transfer function over a period of time in response to the input signal that impinge on the processing element. The adaption is usually acomplished by modification of the values of variables stored in the processing element's local memory [2,3,11,14].

2.2 MODELS OF NEURAL NETWORKS

Artificial neural networks have been developed in a wide variety of configurations [1,2,10,11]. Despite the aparent diversity, network paradigms have a great deal in common. There are two popular models of neural networks : the *feedforward model* and the *feedback or recurrent model*. Depending on the number of layers a neural network may be single layered or multilayered .

2.2.1 The feedforward model

Although a single neuron can perform certain simple tasks the power of neural computation comes from connecting the neurons into networks. In the feedforward model, the neurons are arranged in layers. There are only directed synapses between each layer and the next. Thus the connections are loop free. The inputs are applied in the first layer and the outputs are collected from the last layer. A feedforward network is a special case of combinational circuits with the additional feature that the intermediate variables in the network can assume non-binary values. A two layer feedforward network is shown in Fig.2.2. The circular nodes on the left serve only to distribute the inputs . They perform no computation and hence will not be considered to constitute a layer. The set of inputs X has each of its elements connected to each neuron through a separate weight. Actual artificial or biological network may have many of the connections deleted but full connectivity is shown for reasons of generality .

The output is calculated by multiplying the input vector by the first weight matrix W_1 and then multiplying the resulting vector by the second weight matrix W_2 . This can be expressed as

$$O = (XW_1)\ W_2$$

Since matrix multiplication is associative these terms can be rearranged and written as

$$O = X\ (W_1W_2)$$

2.2.2 Recurrent networks

In the recurrent or feedback model of neural networks connections through the weights extends from the outputs of a layer to the inputs of the same or the previous layer. This special class of neural networks is of considerable interest and widely used.

Nonrecurrent networks have no memory; their output is solely determined by the current inputs and the values of the weights. On the other side, in recurrent network the output is determined by the current inputs and their previous outputs. For this reason recurrent network can exhibit properties very similar to short term memory in human brain, because the state of the network outputs depends in part upon the previous inputs. Fig. 2.3 shows a recurrent network consisting of two layers. The first layer as in previous illustration serves no computational function; it simply distributes the network outputs back to the inputs. Each neuron in the next layer computes the weighted sum of the inputs producing a NET signal that is often operated by a nonlinear function F to yeild the OUT signal.

The important drawback of such recurrent network is that they are not unconditionally stable. Sometimes they enter a mode in which the output wanders interminably from state to state , never producing an usable output. Unstable networks have interesting properties and have been studied as examples of chaotic systems. Recurrent networks using backpropagation and segregation algorithm is discussed later in more details.

2.2.3 Bidirectional associative memory

Human memory is always associative in nature. If we allow our thought to wander they move from topic to topic based on chain of mental association. Binary Associative Memory(BAM) model [10,11] uses this aspect of human brain. BAM accepts the input vector on one set of neurons and produces a related but different output vector on another set. The BAM is capable of generalisation, producing correct output despite of corrupted inputs. These characteristics are strongly reminiscent of human mental functions and bring neural networks one step closer to emulation of brain.

Recent publications have presented several forms of bidirectional associative memories . Fig. 2.4 shows the basic BAM cofiguration. Here an input vector A is applied to the weight network W and produces a vector of neuron outputs B. Vector B is then applied to the transpose of the first weight network W^t which produces new outputs for vector A. This process is repeated until the network arrives at a stable point where neither A nor B is changing. This process can be expressed in symbols as follows :

$$b_i = F(\ a_j w_{ij})$$

or in vector form

$$B = F(AW)$$

where

B = the vector of outputs of layer2

A = the vector of outputs from layer1

W = the weight matrix between layer1 and layer2

F = the activation function

similarly,

$$A = F(BW^t)$$

where

W^t = the transpose of matrix W

2.3 LEARNING STRATEGIES

Learning is the most important properties of neural networks. Neural learning is defined as "the process of adaptively evolving the internal parameters (e.g. connection weights, network topology, etc.) in response to the stimuli being presented at the input and possibly the output buffer" [14]. A wide variety of training algorithms have been developed, each with its strengths and weaknesses. Table I summerizes different types of computational learning paradigms and corresponding algorithms and systems.

Neural network learning procedures can be divided into three broad categories:

(1) Supervised procedures where the desired response is from a knowledgeable teacher and the retrieval involves one or more of a set of stimuli pattern that has been repeatedly shown to the system during training phase.

(2) The unsupervised procedures construct internal models that capture regularities in their input vectors without receiving any additional information.

(3) The reinforcement procedures which requires a single scalar evaluation of the output.

2.3.1 Supervised Learning

TH-6870

Supervised learning requires the pairing of each input vector with a target vector representing the desired output. Such a pair is called *training pair*. Usually a network is trained over a number of such training pairs. The network observes the presented inputs, detects the statistical regularities embedded within it and learn to exploit these regularities to draw conclusions when presented with a portion or distorted version of the original pattern. When the portion of the original pattern is used as a retrieval cue, the learned process is denoted to be *auto-associative*. When the desired input is different from the actual input, then the learning is reffered as *hetero-associative* [13].

2.3.2 Unsupervised learning

Despite many application successes supervised learning has been critisized as being biologically implausible. It is difficult to conceive of a traning mechanism in the brain that compares desired and actual outputs , feeding processed corrections back through the network. Unsupervised training is far more plausible model of learning in the biological system. Such a learning scheme was firstly introduced by Kohonen and later by others. Unsupervised learning does not require target vector for the outputs and hence, no comparisions to predetermined ideal responses. The training set contains solely input vectors. The traning algorithm modifies the network weights to produce output vectors those are consistent. The traning process therefore extracts the statistical properties of the training set and groups similar vectors into classes. Appliing a vector from a given class to the input will produce specific output vector, but there is no way to determine prior to training which specific output pattern will be produced by given input vector class.Hence the outputs of such a network must generally be transformed into a comprehensible form subsequent to the training process. This does not represent a serious problem . It is usually simpler matter to identify the input-output relationships established by the network.

2.3.3 Reinforcement learning procedures

The central idea in the reinforcement learning procedure is that, we can assign credit to a local decesion by measuring how it correlates with the global reinforcement signal. Various different values are tried for each local variable(such as the state or a weight) and these variations are correlated with variations in the global reinforcement signal. Normally, the local variations are the result of independent stochastic processes [12]. So if enough samples are taken, each local varible can average away the noise caused by the variations in other variables to reveal its own effect on the global reinforcement signal. The network can then perform gradient ascent in the expected reinforcements by altering the probability distribution of the value of each variable in the direction that increases the expected reinforcement . If the probability distributions are altered after each trial, the network performs a stochastic version of gradient ascent.

2.4 INADEQUACIES OF STATIC LEARNING

One of the major strengths of neural network is their ability to recognise or correctly classify patterns which have never be presented to the network before . Neural networks appears unique in their ability to extract the essential featurs from a training set and to use them to identify new inputs. In any network the topology of the network(i.e. the number of hidden layers and the number of neurons in different layers) is crucial not only to the ability of classifying training data, especially to the network's ability to generalise. Generalisation means selecting a curve that is a model of the properties of the data. Since the curve can only be modelled on the training data, the training set must be an all encompassing subset of the entire data set. Any parameters of the model of the entire data set those are not represented in the training set cannot be accurately modelled. Since the entire data set is usually unknown, the quality of the network's internal model depends upon the accuracy of the designer's guesses. So in a static learning procedure , to produce meaningful results it requres an adequate network topology. The essential problem in modelling is finding a topology that already models the data [6].

The number of hidden layers and their interconnection depends on the complexity of the learning task;more hidden units are required to approximate a more complex function. Over learning occurs when the number of connections reaches the number of training patterns, that is every connection is dedicated to one specific pattern. This also reduces the network's ability to generalise, that is the patterns of the test set perform significantly worse than patterns of the training set , because each patern is trained separately and no common features are represented in the network's internal model. Just by increasing the number of hidden units and their interconnections does not necessarily lead to an adequate representation of the features of the data set; rater the simulation time increases. The order of magnitude of the increase of simulation time is almost squre. Sometimes static learning processes get trapped in a local minimum [7,11,16] and also suffer from many other problems discussed in the following chapter.

2.5 NON-STATIC LEARNING AND SELFORGANISATION

In the problem domain where the number of possible patterns is unlimited and the number of classes are not priori known, it may be impossible to estimate the number of required hidden units in advance. Moreover, the distribution of patterns over time may vary. To allow adaption to the changing situation the number of hidden units must change. Hence, the training algorithm should be designed such that not only the weights are modified but also overall topology of the network changes depending on the complexity of the class of problem. Non-static learning is one such kind of higher structured learning procedure.

Non-static neural networks follows the dynamic aspects of biological network. It has been shown that in the human brain whenever a group of neurons is working on a task which be cannot accomplished by them because they are too few in number,then another bunch of neuron cells get devoted to this task in order to support the others. Taking into the consideration the importance of network topology and biological evident dynamic topology with respect to a specific task, non-static neural networks are introduced.

A non-static paradigm is defined as *changing the topology of the network not only during the learning phase but also at every time after its static definition which precedes learning or even as allowing to define the ovrall topology dynamically, while learning*. In a dynamic paradigm units can be created at every time, before,during or after learning. They can be connected arbitarily with other units of the network [6,16].

2.6 PRUNING IN NEURAL NETWORKS

In the above discussed approach to neural network design we start with a network having a hidden layer with very small number of units and lead to a larger network which is able to differentiate classes containing many representatives. On the other hand, *pruning* is almost the opposite approach of examining a solution network. Here the process starts with a reasonably larger neural network and the units which are not necessary to the solution is removed step by step. In this process the network is analysed by examining the outputs of the hidden units across all the training set inputs. There are two stages of pruning.

One is removing units that can be considered as not contributing to the solution. Stage two is removing units that give information not requried at the next layer [7].

2.6.1 Removing noncotributing units :

These are units which eithor have approximately constant output across the training set or have outputs across the training set which mimic the outputs of another unit. These units can be removed and the weights given to their outputs are redistributed in such a way that there will be almost no change to the network performance over the training set.

2.6.2 Removing unnecessary information units :

This is the second stage of pruning where the units which are independent of other units in the layer but give information which is not required at the next layer is removed. Removing units which contribute unnecessary information to the next layer may lead to the outputs of the pruned layer being linearly inseparable with respect to the classes or outputs of the above layer.

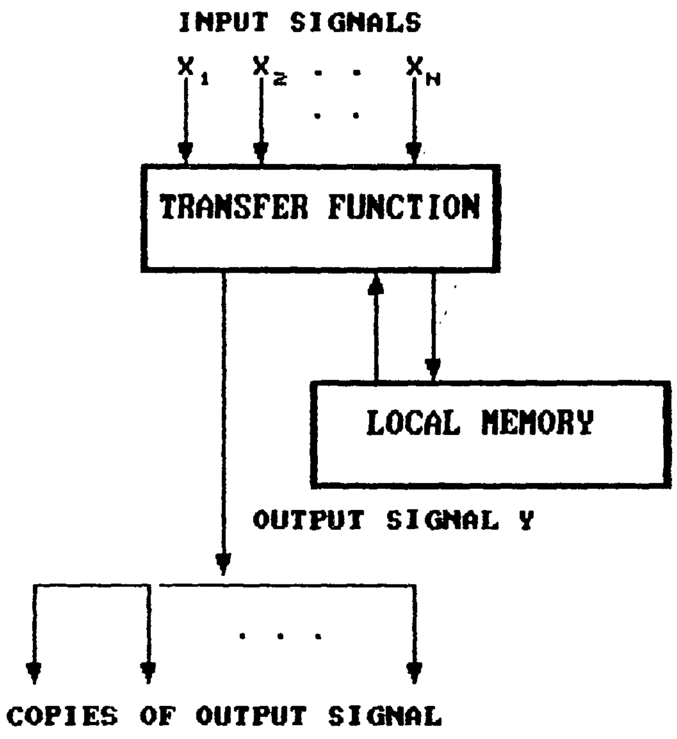


FIG.2.1 INTERNAL DETAILS OF A PROCESSING ELEMENT

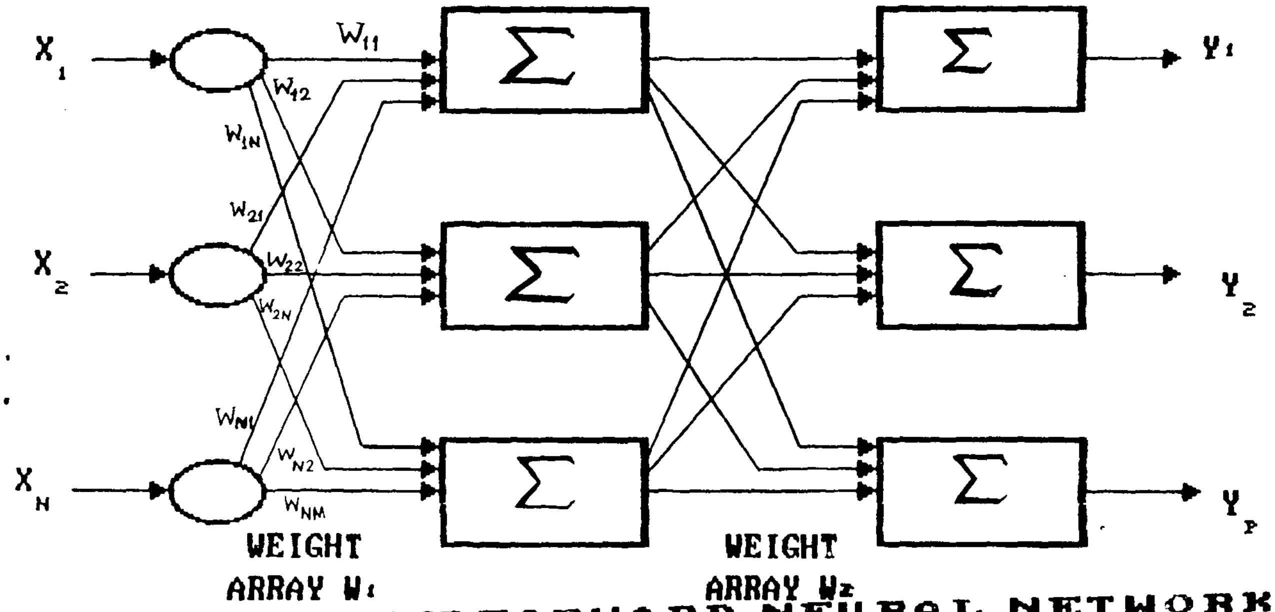


FIG.2.2 FEEDFORWARD NEURAL NETWORK

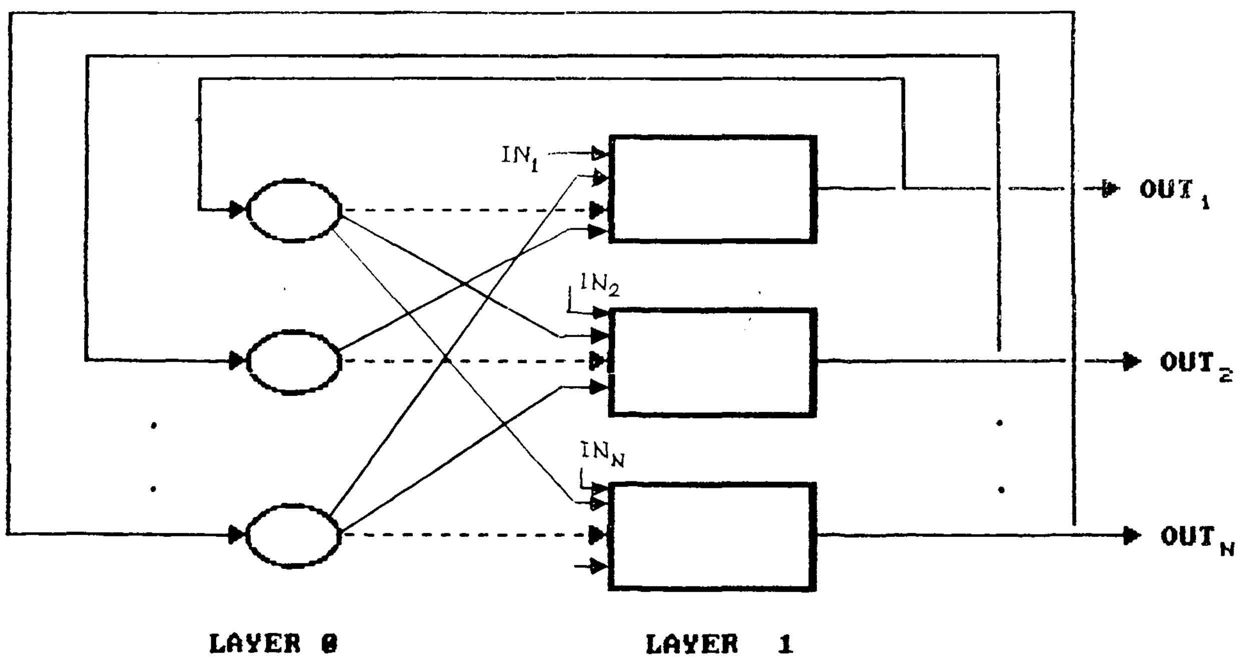


FIG.2.3 A MODEL OF RECURRENT NEURAL NETWORK

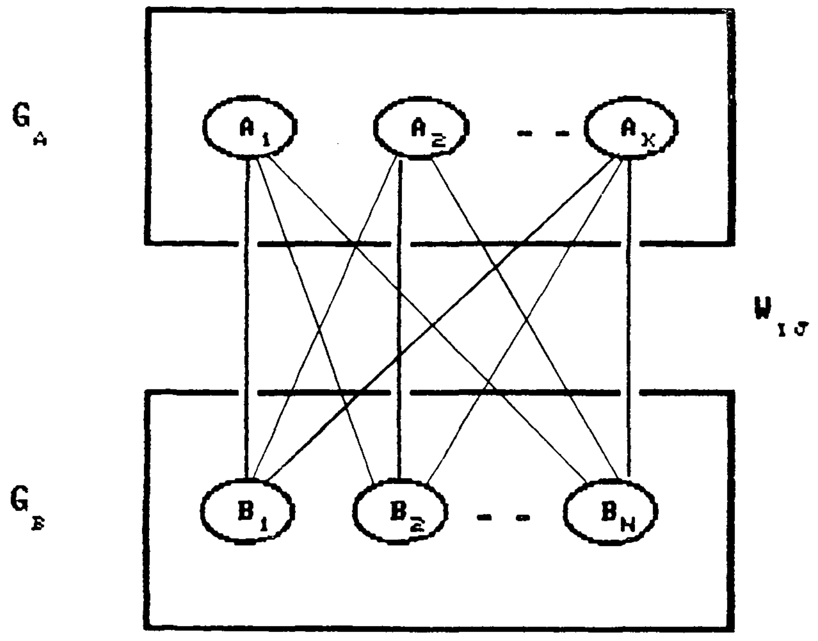


FIG.2.4 BIDIRECTIONAL ASSOCIATIVE MEMORY

Paradigm	Categories	Algorithm description	Nature	Systems
Empirical learning	Rule-based	Real-to-model world mapping	Deductive	LAMP, PRODIGY
	Learning via query	Ordering of questions directs learning	Inductive	MARVIN
	Concept learning	Search in hypothesis space	Inductive	OTIS
Case-based learning	Adapting case bases	Search through case networks of analogy-explanatory-emulation skills	Deductive	TA
Statistical learning	Bayesian tech., decision trees	Find probable number of classes, probabilistic descriptions, and probability of belongingness	Inductive	AutoClass
Genetic learning	Population genetics	Trial solutions (populations) operated in cycles (generations) by survival-of-fittest selection followed by genetic recombination (crossover and mutation operators)	Inductive	CFS
Explanation-based learning	Reasoning about operationality (merit of a learning result)	Obtain general concept definition of some property that holds for a given training instance	Deductive	ROE, MetalLEX
Connectionist (neural) learning	Supervised, unsupervised, reinforcement	Capture the functional, spatial, or temporal concept in the internal synaptic connections or topology of the network	Relaxation in an energy landscape	EBP, CL, ART, SID
Distribution-free learning	Formal concept learning	Construct a minimal set of maximally general descriptions	Inductive, Deductive	PAC

TABLE 1

CHAPTER 3

NON-STATIC LEARNING ALGORITHM

In recent years major thrust has been given on the study of nonlinear dynamic systems theory. As a result many enhancements has been proposed on the conventional static learning algorithms for neural networks. Some AI researchers have shown tremendous interest in designing dynamic neural networks which adapts to the environment. In this chapter the basic backpropagation algorithm has been discussed in detail. Also an efficient and adaptive learning algorithm has been presented which trains the network to decide its optimal size by itself for a particular class of problem. Finally the *segregation algorithm*, a more advanced version of non-static algorithm is also discussed.

3.1 THE BACKPROPAGATION ALGORITHM

The backpropagation algorithm is the most suitable supervised learning algorithm for multilayer neural networks. The invention of backpropagation has played a large part in the resurgence of interest in artificial neural networks. A clear and concise definition of backpropagation algorithm was first presented by Rumelhart, Hinton and Williams in 1986. Later it was popularised by Rumelhart in the Parallel Distributed Processing(PDP) edited volume in 1980. The PDP researchers suggested that the backpropagation algorithm or as they called it *generalised delta rule* is the most suitable learning algorithm for neural networks and overcome the limitations of the perceptron algorithm [1,3,11,16].

3.1.1 Network cofigurations

Neurons are the fundamental building blocks of backpropagation networks. A set of inputs is applied eithor from the outside or from a previous layer. Each of these is multiplied by a weight and the products are summed. This summation of products are termed as NET and must be calculated for each neuron in the network. After NET is calculated, an activation function F is applied to modify it, thereby producing a signal OUT. Fig 3.1 shows the artificial neuron with activation function. Fig 3.2 shows the sigmoidal activation function which is usually used with backpropagation. The OUT signal can be represented by

$$\mathrm{OUT} = 1/(1+e^{-\mathrm{NET}}) \tag{3.1}$$

The sigmoid function, also called a logistic or a squashing function is desirable because it has a simple derivative . The derivative of the OUT function can be expressed as

$$\frac{\partial \mathrm{OUT}}{\partial \mathrm{NET}} = \mathrm{OUT}(1-\mathrm{OUT}) \tag{3.2}$$

The sigmoid function compresses the range of NET so that OUT lies between zero and one. As discussed previously, multilaer networks have greater representational power than single layer network only if a nonlinearity is introduced. The sigmoid function produces the required nonlinearity. Any nonlinear function other than the sigmoid function can be used, which sould be differentiable everywhere. The sigmoid function has the additional advantage of providing a form of automatic gain control. The shape of the function is such that large signals can be accomodated by the network without saturation, small signals are allowed to pass through without excessive attenuation.

Fig. 3.3 shows a multilayer network that is suitable for training with backpropagation algorithm. The first set of neurons serve only as distribution points; they perform no summation. The input signal is simply passed through to the weights on their outputs. Each neuron in the subsequent layer produces NET and OUT signal as described above.

3.1.2 An overview of training

Training is the process of adjusting the weight so that application of a set of inputs produces the desired set of outputs. The input-output sets can be reffered as vectors. Training assumes that each input vector is paired with target vector representing the desired output; together they are called *training pair*. The network is trained over a number of training pairs. For example, the input part of a training pair might consist of patterns of ones and zeros representing a binary image of an alphabet. The output may also be a set of binary numbers which represents the letter.

Before starting the training process, all the weights must be initialised to small random numbers. This assures that the network is not saturated by large values of the weights and prevent certain other training pathologies.

Training the backpropagation network requires the following steps :

1. Select the next training pair from the training set and apply the input vector to the network input.
2. Calculate the output of the network.
3. Calculate the errors between the network output and the desired output.
4. Adjust the weights of the network in such a way that the error is minimized.

5. Repeat step 1 through 4 for each vector in the training set until the error for the entire set is acceptably low.

Calculations are performed layer by layer basis. Referring to the Fig. 3.3 , first the outputs of the neurons in the layer j is calculated and these are used as input to the layer k. The output of the neurons in layer k is calculated which constitute the network output vector.

It may be seen that in the steps 1 and 2 the signal propagate from the network input to the network output. So this is called as the *forward pass*. In steps 3 and 4 the error signal propagates from the output to the input through the hidden layers where it is used to adjust the weights. This is called as *reverse pass*. The two passes are elaborated below .

3.1.3 Forward pass

The above said forward pass can be expressed in a more mathematical form as follows: The input vector pair X and the target vector T comes from the training set. The calculation is performed on X to produce the output vector Y. Calculations in a multilayer network are done layer by layer starting at the layer nearest to the inputs. The NET value of each neuron in the first layer is calculated as the weighted sum of its neurons' inputs. Then the activation function F operates on the NET to produce the OUT value for each neuron in that layer. Once the set of outputs for a layer is found then it serves as an input to the next layer. The process is repeated layer by layer until the final set of network outputs is produced.

In vector notation,

$$N = (XW) \tag{3.3}$$

where N = the NET vector

W = the weight matrix

Applying function F to the NET vector N, component by component the outputvector O is produced.

$$O = F(XW) \tag{3.4}$$

The output vector for one layer is the input vector for the next. So the equation 3.4 is applied to each layer to calculate the outputs of the final layer.

3.1.4 Reverse pass

In this pass, the error between the desired output i.e. the target and the actual output is obtained from the forward pass is calculated and subsequently used to adjust weights of the output layer. Fig. 3.4 shows the training process for a single weight from neuron p in the hidden layer j to the neuron q in the output layer k. The output of a neuron in the layer k is subtracted from the target value T to produce the ERROR signal. This is multiplied by the derivative of the squashing function [OUT(1-OUT)] calculated for the layer's neuron k, thereby producing the δ value which is expressed as

$$\delta = OUT(1-OUT)(T-OUT) \tag{3.5}$$

The s is multiplied by OUT from neuron j and a training rate coefficient r and the result is added to the initial weight to get the modified weight.

$$\Delta w_{pq,k} = r\,\delta_{q,k}\,OUT_{pj} \tag{3.6}$$

$$w_{pq,k}(n+1) = w_{pq,k}(n) + \Delta w_{pq,k} \tag{3.7}$$

where $w_{pq,k}(n)$ = the value of a weight from neuron p in the hidden layer to the neuron q in the output layer at step n before weight adjustment

$w_{pq,k}(n+1)$ = value of the weight at step (n+1) after adjustment

$\delta_{q,k}$ = the value of s for neuron q in the output layer k

OUT_{pj} = the value of OUT in neuron p in the hidden layer j

Since the target value is available for each neuron in the output layer, adjusting the associated weights is easily accomplished. But the hidden layers have no target vector. So the hidden layers are trained by backpropagating the output error through the network layer by layer, adjusting weights in each layer. Eq. 3.6 and 3.7 are used for all layers, both output and the hidden layers. First, δ is calculated for each neuron in the output layer using Eq.3.5. After using it to adjust the weights in the output layer, it is propagated back through the same weights to generate a value of δ for each neuron in the hidden layer adjacent to the output layer. These values of δ are used to adjust the weights of the hidden layer and in a similar way, are propagated back to all preceding layers.

Consider a single neuron in the hidden layer just before the output layer. In the forward pass, the neuron propagates its output to neuron in the output layer through the interconnecting weights. During the reverse pass, these weights operates in reverse, passing the value of δ from the output layer back to the hidden layer.Each of these weights are multiplied by the δ value of the neuron to which it connects in the output layer. The value of s needed for the hidden layer neuron is produced by summing all such products and multiplying by the derivative of the squashing function :

$$\delta_{pj} = OUTpj(1-OUT_{pj})\ (\Sigma\delta_{q,k}w_{pq,k}) \qquad (3.8)$$

For each neuron in a hidden layer, s must be calculated and all weights associated with that layer must be adjusted. This is repeated moving back towards the input, layer by layer until all weights are adjusted. In a vector form it can be expressed as

$$D_j = D_k W_k{}^t \ \$ \ [O_j \$ (I-O_j)] \qquad (3.9)$$

where,

D = the set of ss(subscript denotes the layer)

\$ represents the componentwise multiplication

I = Unit vector whose all elements are 1

O_j = Output vector of layer j

W_k = Wight matrix of layer k

In many cases, each neuron is presented with a trainable bias, so that the training process converges more rapidly.

3.1.5 Momentum

Momentum is a method of improving the training time of the backpropagation algorithm, while enhancing the stability of the process. This involves adding a term to the weight adjustment that is proportional to the amount of previous weight change. So the weight adjustment equations can be modified as

$$\Delta w_{pq,k}(n+1) = r(\ _{q,k}\ OUT_{pj}) + \alpha\,[\Delta w_{pq,k}(n)] \quad (3.10)$$

$$w_{pq,k}(n+1) = w_{pq,k}(n) + \Delta w_{pq,k}(n+1) \quad (3.11)$$

where α is the momentum coefficient which is taken around 0.9

3.2 ENHANCEMENTS IN BACKPROPAGATION ALGORITHM

Many promising developments and extentions has been done to the basic backpropagation algorithm in the last few years [11]. In 1987, Parker described a method for increasing the speed of convergence of the backpropagation algorithm called as *second order backpropagation*. It uses second derivatives to produce a more accurate estimate of the correct weight change. Later, Stornetta and Huberman described a relatively simple method for improving the training characteristics of backpropagation networks.They suggested the change of the dynamic range of inputs and hidden neuron outputs from (0to1) to $\pm$ 1/2. The new squashing function is as follows

$$OUT = -1/2 + 1/(e^{-NET}+1) \quad (3.13)$$

By this method the convergence time reduced by 30 to 50%. In 1988, Pineda and Almeida described methods for applying backpropagation to recurrent networks. They showed that learning can be occured very rapidly in such systems and the stability criteria is easily satisfied.

3.3 DEMERITS OF STATIC BACKPROPAGATION ALGORITHM

Backpropagation has been applied to a wide variety of research applications. Despite the demonstrated power of backpropagation, several problems plague its application. Most troublesome is the long uncertain training process. Sometimes the training suffers from total failure due to different causes as described below :

3.3.1 Convergence

Rumelhart, Hinton and Wlliams has provided a convergence proofcast in terms of partial differential equation, making it valid only if network weights are adjusted in infinitesimal steps. Because this implies infinite convergence time, it proves nothing about the training time in practical applications. In fact, there is no proof that the backpropagation will ever converge with a finite step size.

3.3.2 Local minima

Backpropagation employs gradient descent to adjust the network weights following the local shape of the error surface towards the minimum. This works well with convex error surfaces, which have a unique minimum, but it often leads to nonoptimal solutions with highly convoluted, nonconvex surfaces encountered in practical problems. In such cases the network is trapped in a local minimum. This is shown in Fig. 3.5 . Even after the network is trained, there is no way to tell if the backpropagation has found the global minimum or not. If the solution is not satisfactory, one is obliged to initialize the weights to new random values and to retrain the network, with no guarantee that it will train on the given trial and that a global minimum will ever be found.

3.3.3 Network paralysis

Sometimes, as the network trains, the weights are adjusted to very large values. This can force all or most of the neurons to operate with large values of OUT in the region where the derivative of the squashing function is very small. Since the error sent back for training is proportional to this derivative, the training process can come to a virtual standstill. There is no theory that predicts whether or not a network will be paralysed during training.

3.4 THE PROPOSED NON-STATIC ALGORITHM

Backpropagation is often used as the learning algorithm in layer structured neural networks because of its efficiency. But the optimum number of hidden layers and the number of neurons in each layer is not known in advance. It has to be decided eithor randomly or by hit and trial. So the network's performance depends on designer's guess. So an algorithm has been proposed where we start with a relatively small network and systematically lead to a network with optimal number of hidden units.

The proposed algorithm can be summerized by the following steps :

1. Initialise the network with small number of hidden layer and each with a small number of neurons.

2. Train the network using conventional backpropagation algorithm with first few training sets and get a temporary optimal weight matrix.

3. Apply the rest training sets and get the sum of difference vectors i.e. the deviation of the output vector from the target vector.

4. Calculate the average value of error per neuron in the output layer. i.e.

$$\text{error} = (\sum_{i=1}^{i=q} D_i)/(q*\text{no.of training set}) \qquad (3.14)$$

5. Compare the current error with the minimum error. (The minimum error is initialised to 1 which is the maximum possible value)

6. If the current error is less than the minimum error, then assign the current value of error to minimum error. Increse the size of the network and go to step 1.

7. If the current error is not smaller than the minimum error, then print the optimum size of the network and stop.

An assumption has been taken that whenever the performance of the network degrades, in comparision to the smaller network in the previous step, the performance will not improve by incresing the size of the network still further. This has been observed during implementation. There are two different approaches to increse the size of the network as mentioned in the step 6 of the algorithm. According to the first approach, for each fixed number of neurons, the number of layers are incresed until the minimum error termed as minerror(l) reaches its first minimum. Then the number of neurons are fixed at the next higher level and the number of layers is again incresed. This whole process is repeated until we get the first overall minimum called as minerror(opt). Finally, the optimum size of the network corresponding to the overall minimum error is reported.

In the other approach, we increse the size of the network in a different manner. Here, the number of layers is first fixed and the number of neurons in different layers is increased to get the first minimum error termed as minerror(n). Then the number of layers is increased one by one until we get the overall minimum error minerror(opt) as in the first approach. Here also the optimum size of the network is reported. These two approaches lead to the same results but with different directions. These approaches are described by flowcharts given in Flowchart 1 and the Flowchart 2.

3.5 THE SEGREGATION ALGORITHM

The segregation algorithm is a higher structured non-static algorithm proposed by F. Schonbaur and M. Kohle in 1986 [6]. This algorithm overcomes almost all demerits of the static backpropagation algorithm.

As mentioned previously, the ability to generalise using backpropagation algorithm is very sensitive to the number of hidden units. The separation into classes learned by the network after a sufficient number of training steps is describable by a simple rule since those few hidden units involved in the classification task could only differentiate between the majority of input patterns and the rest. If learned patterns are removed from the training set, the remaining patterns can be used as training set for another network that will have to deal with the classes that are left. This segregation of patterns and their assignment to their own network can be repeated as often as necessary.

The algorithm can be formulated as follows :

1. Use backpropagation learning algorithm with small number of units in the hidden layer until the mean squared error of a certain percentage of patterns is significantly less than the average mean squared error for all patterns.
2. Continue learning using these patterns until the error approaches zero.
3. Remove these patterns from the training set.
4. Clamp all weights conected to or from the hidden

units.

5. Add a few units to the hidden layer and connect them.

6. If there are still patterns left in the training set, repeat all previous steps.

7. Unclamp all weights

8. The weights should now represent a good initial position on the error surface, continue normal learning to reunite all subnets.

With this algorithm classes that are difficult to learn because there are so few patterns for these classes in the training set can be learned separately. They can be presented as often as desired to a specific subnetwork without disturbing the knowledge of the rest of the network that has already learned the other classes. The robostness of the neural networks to noisy data or very infrequent apperances of stray patterns can be explicitly weakened to any desired degree without entirely removing this feature.

The algorithm is sensitive to its parameters, namely the percentage of patterns that has to have a different error rate to be recognised as a different entity and learned separately, the number of hidden units the network starts with , the number of hidden units that is added at each segregation step.

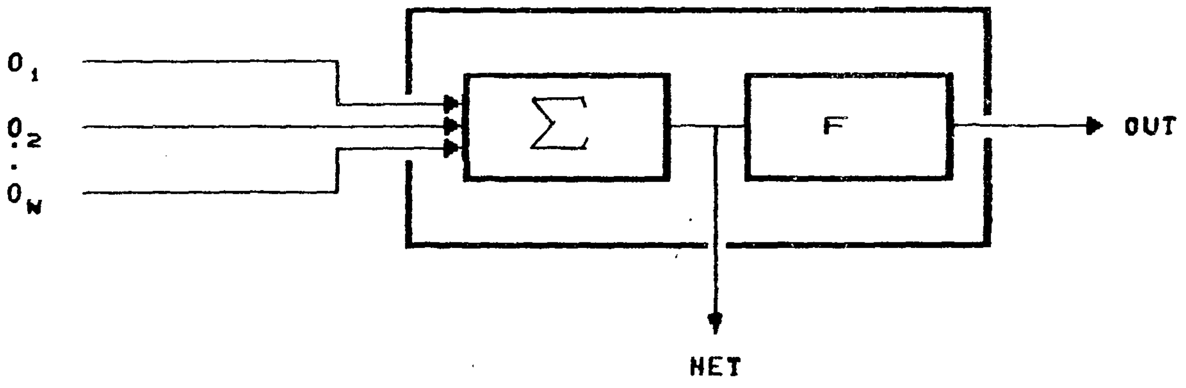


FIG.3.1 ARTIFICIAL NEURON WITH ACTIVATION FUNCTION

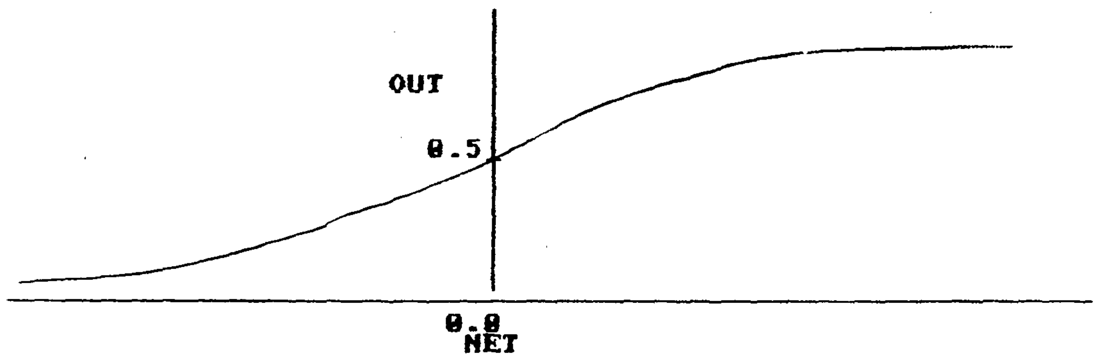


FIG.3.2 SIGMOIDAL ACTIVATION FUNCTION

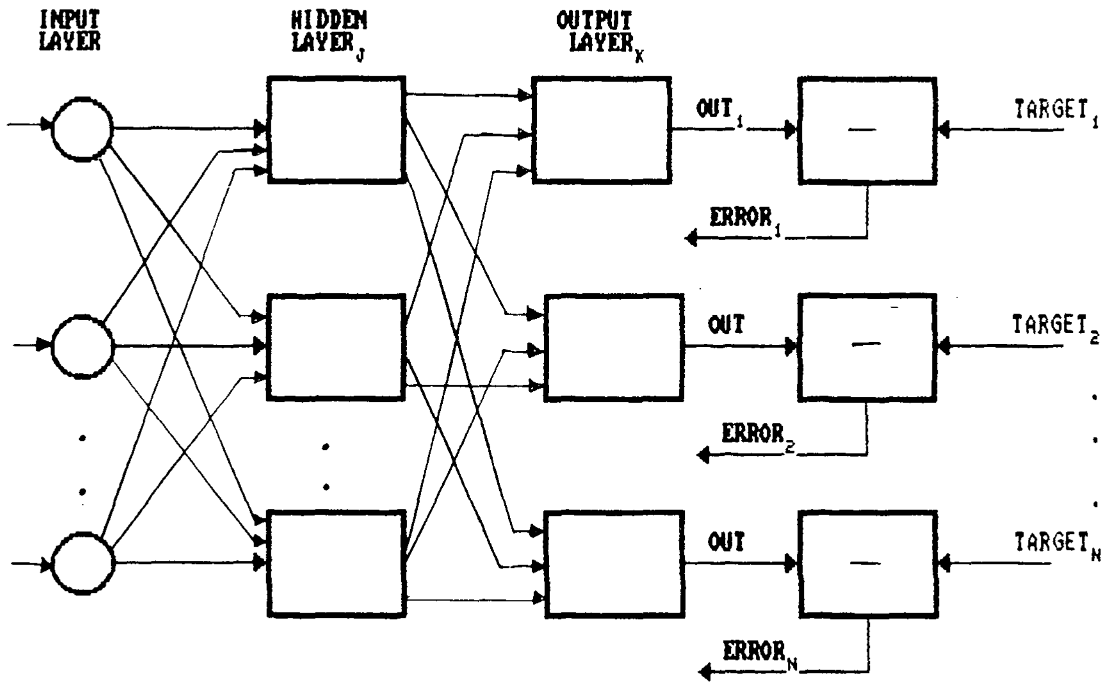


FIG.3.3 A TWO-LAYER BACKPROPAGATION NETWORK

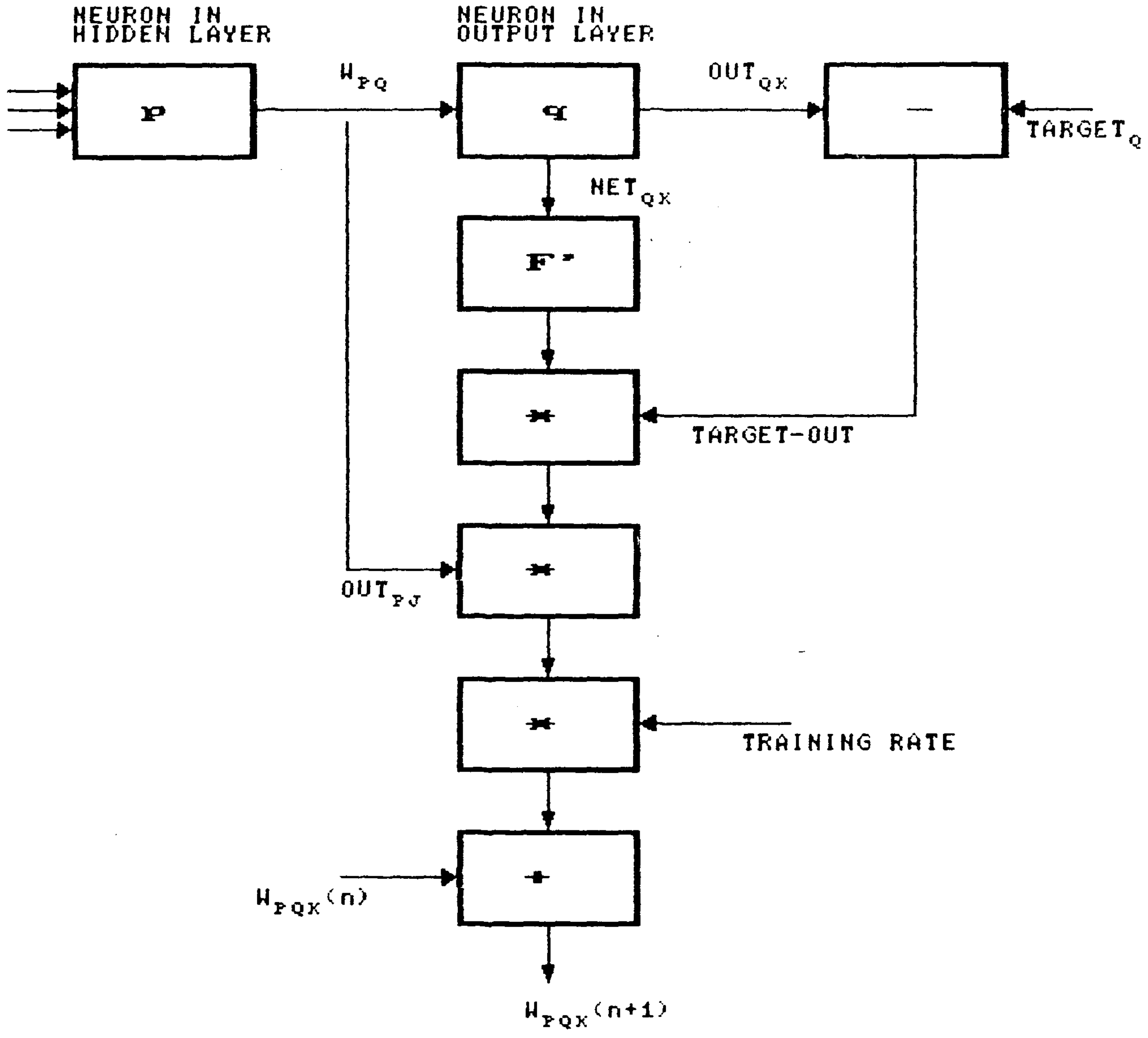


FIG.3.4 MODIFYING A WEIGHT IN THE OUTPUT LAYER

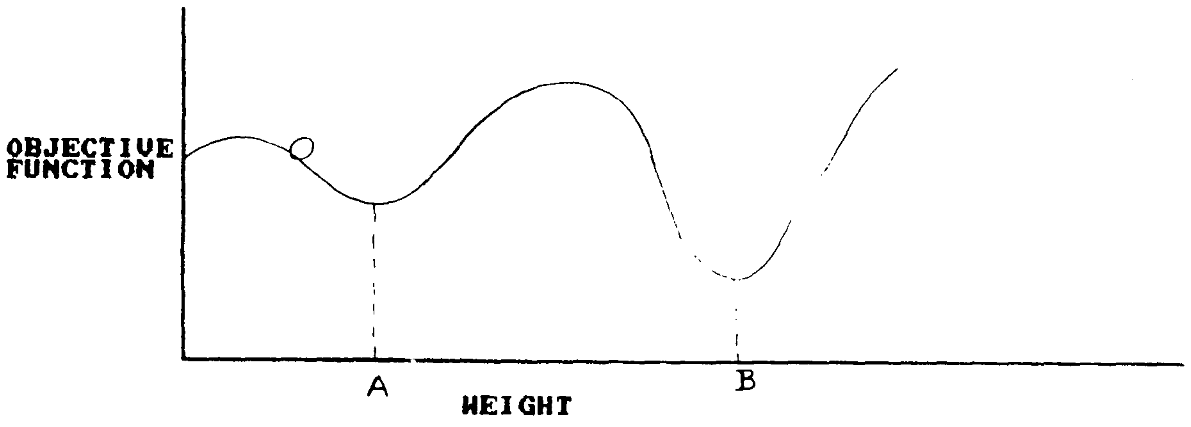
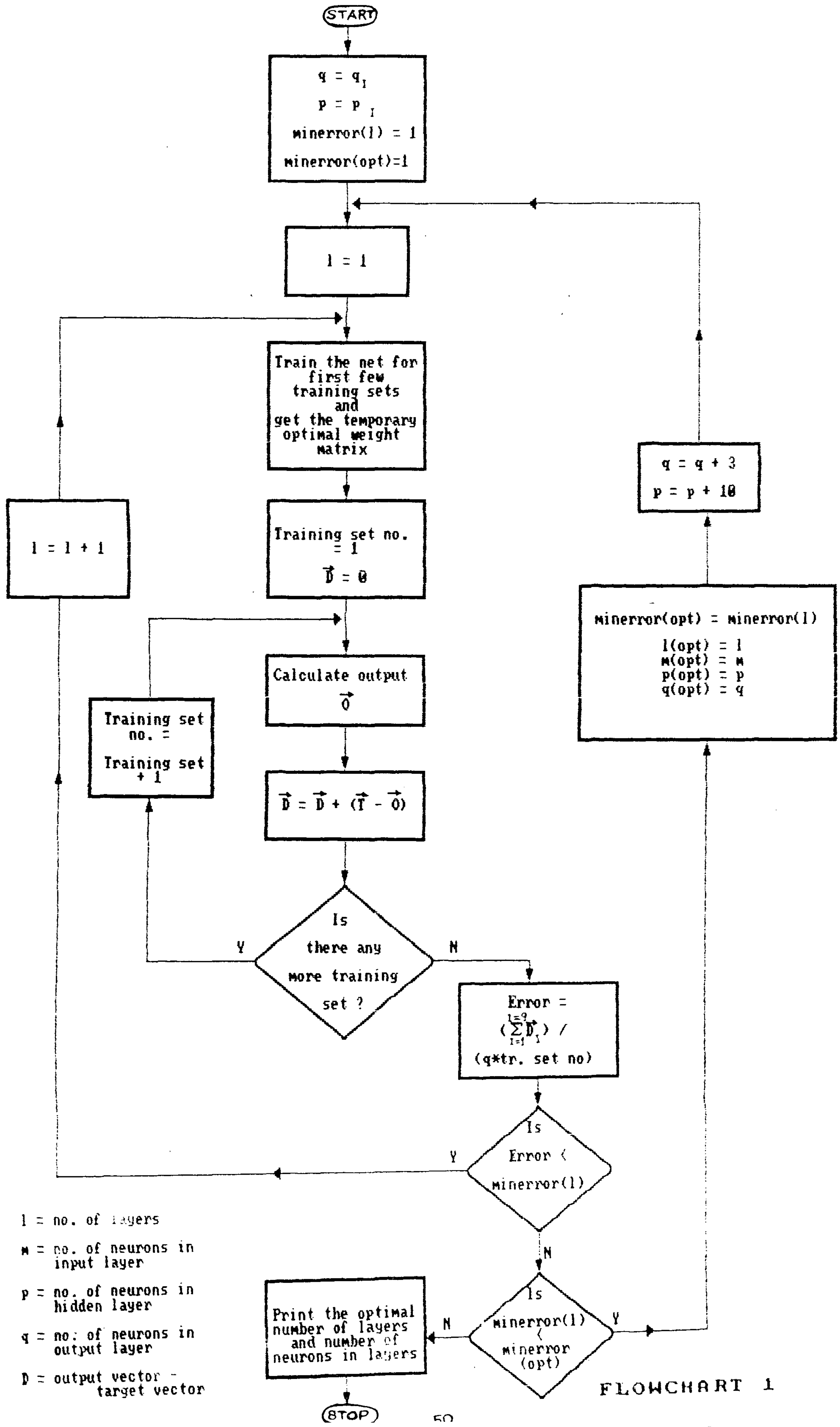


FIG.3.5 LOCAL MINIMUM PROBLEM


START
q = q_I
p = p_I
minerror(l) = 1
minerror(opt)=1
l = 1
Train the net for first few training sets and get the temporary optimal weight matrix
Training set no. = 1
D = 0
Calculate output O
D = D + (T - O)
Is there any more training set ?
Y
N
Training set no. = Training set + 1
Error = (ΣD_l) / (q*tr. set no)
Is Error < minerror(l)
Y
N
l = l + 1
Is minerror(l) < minerror (opt)
Y
N
minerror(opt) = minerror(l)
l(opt) = l
m(opt) = m
p(opt) = p
q(opt) = q
q = q + 3
p = p + 10
Print the optimal number of layers and number of neurons in layers
STOP

l = no. of layers

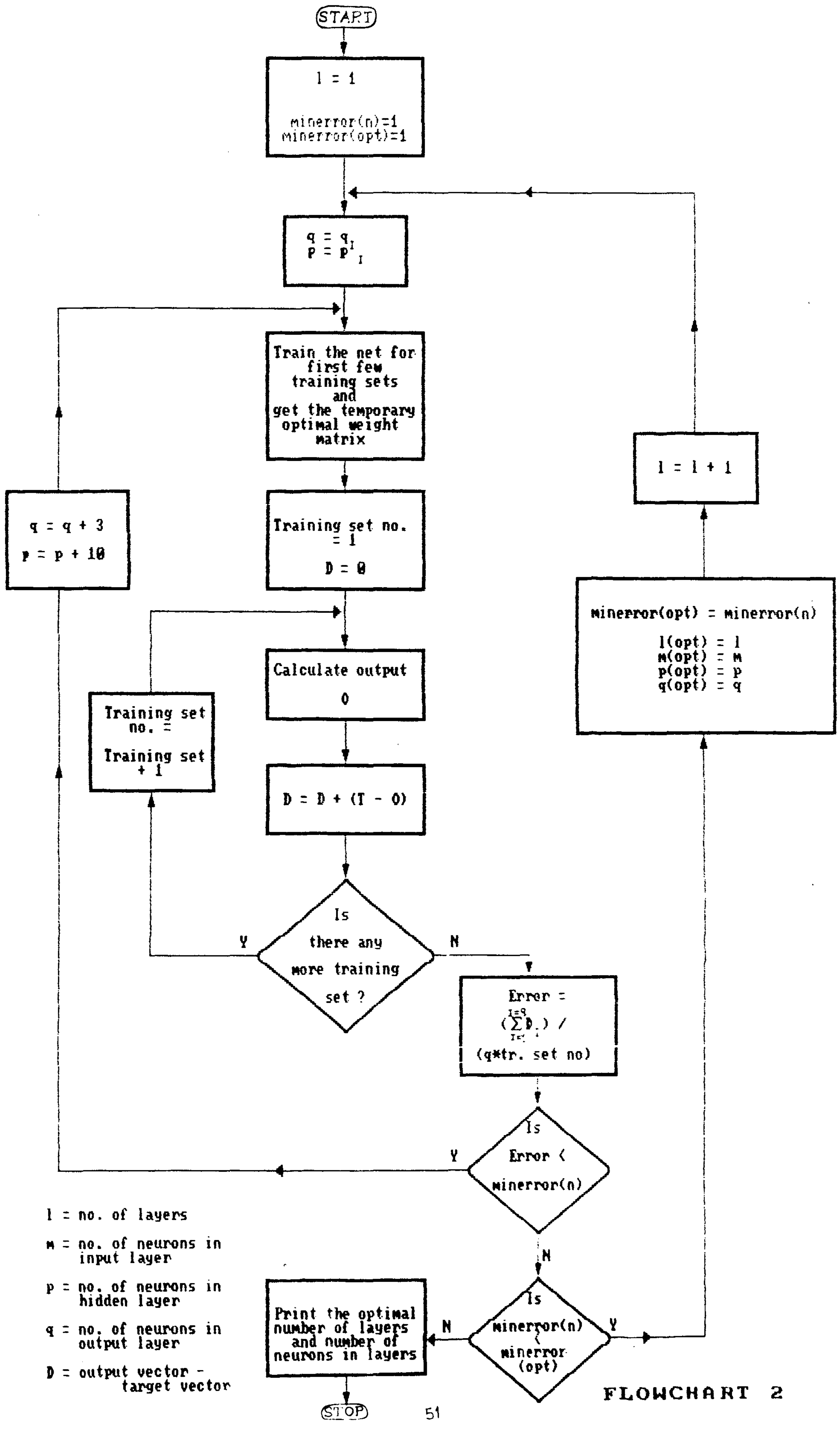
m = no. of neurons in input layer

p = no. of neurons in hidden layer

q = no. of neurons in output layer

D = output vector - target vector

FLOWCHART 1


START
l = 1
minerror(n)=1
minerror(opt)=1
q = q_I
p = p_I
Train the net for first few training sets and get the temporary optimal weight matrix
Training set no. = 1
D = 0
Calculate output O
D = D + (T - O)
Is there any more training set ?
Y
N
Training set no. = Training set + 1
q = q + 3
p = p + 10
Error = (ΣD_i) / (q*tr. set no)
Is Error < minerror(n)
Y
N
Is minerror(n) < minerror (opt)
Y
N
minerror(opt) = minerror(n)
l(opt) = l
m(opt) = m
p(opt) = p
q(opt) = q
l = l + 1
Print the optimal number of layers and number of neurons in layers
STOP

l = no. of layers

m = no. of neurons in input layer

p = no. of neurons in hidden layer

q = no. of neurons in output layer

D = output vector - target vector

FLOWCHART 2

CHAPTER 4

IMPLEMENTATION AND RESULTS

In this chapter, the implementation of a multilayer neural network, simulated to recognise handwritten characters using the proposed non-static algorithm has been discussed. Also the relationship between network structure and the ability of the network to generalise from the training set has been investigated. At last, the obtained results, given in a tabular form has been analysed.

4.1 NEURAL NETWORKS AND HANDWRITTEN CHARACTER RECOGNITION

The recent resurgence of interest in neural networks, machine learning and parallel computation has led to a renewed research in the area of pattern recognition [8,9,12,15] . Especially, the problem of handwritten character recognition has invoked great research interest for a very long time. There are many difficulties in handwritten character recognition because of the presence of large degree of variations of the data. Not only there exists some changes and distortions of characters from one individual to another, but also there are some variations from the same individual at different times. Furthermore, difficulties may result from problems such as complexity of characters, similarity of different characters etc. The traditional method for recognition of handwrtten characters using statistical techniques require large amount of data to be stored. A more recent and appealing approach is the neural network technique.

Pattern recognition problems require mainly two processes : analyses or feature extraction and classification.

4.1.1 Feature extraction

The analyser receives an input pattern which is a very complex event such as optical signals, speech signals, electrical signals etc.. The analyser extracts the main features of the pattern. Its output is the pattern vector.

The process of analyses or feature extraction of a pattern is generally done by complex hardware. But in this case it has been done manually. The letters are written on a graph paper containing squares with 8X8 and 8X6 boxes . If any portion of the letter is passed through a box then that particular input is taken as one ; otherwise zero. Likewise we analyse the letter and get the corresponding input vector X for each letter. Fig.4.1 and Fig. 4.2 indicates how a character is analysed using 8X6 and 8X8 grids respectively. The data for all samples of handwritten characters is stored in an input file.

4.1.2 Classification

Several neural networks has been proposed for classification problem. Unlike traditional classifiers which tend to test comparing hypothesis sequentially, neural networks test it in parallel, thus providing high computational rates. Here the size of the network is varied to get a network that can classify most efficiently. The minimum number of neurons in the output layer is taken as three. So the network can distinguish eight characters.

4.2 IMPLEMENTATION OF THE NON-STATIC ALGORITHM

A multilayer neural network classifier has been simulated by a program in C language which uses the non-static algorithm. Here the fist approach (discussed in chapter 3) is used where the number of layers is increased for each fixed number of neurons in different layers and finally we get the optimum size of the network.

4.2.1 The network

The process starts with a network having an input, one hidden and the output layer. Two cases has been considered; one with input layer having 48 binary inputs where it classifies the letters written inside the rectangle with 8X6 boxes and the other, where the input layer has 64 units to recognise the characters written in squares with 8X8 boxes. In each case, the initial number of neurons in the output layer is 3. In the first step, the number of neurons in the hidden layer is fixed at 48X3. The elements of the weight matrix is initialised to any random number between -1 to +1. Subsequently, the second and third layer is introduced.

4.2.2 The learning process

To train the network efficiently, thirty training samples are taken in each case. Using the conventional backpropagation algorithm the neural network is trained with the first twenty training sets to get the temporary optimal weight matrix. Then the rest ten training sets are used to calculate the average error per neuron in the output layer. In this case,

$$\text{error} = \left(\sum_{i=1}^{i=q} D_i \right) / q * \text{No. of training sets}$$
$$= \left(\sum_{i=1}^{i=q} D_i \right) / 3*10$$

Then the number of layers in the network is increased to two and the same process is repeated . Whenever the error, termed as minerror(l) in these processes where the number of neurons are fixed and the number of layers are increased, reaches its first minimum, the overall minimum error minerror(opt) is assigned the value of the current minerror(l) and then the number of neurons is increased further to next fixed level. Again the process of increasing the number of layers is done and so on. Finally, when the minerror(opt) reaches its first minimum, the training process is terminated and the optimal number of hidden layers and the number of neurons in different layers is reported.

4.2.3 The program

The program is written in C language which has mainly two sections. The first section of the program is used to train the network using the first twenty training sets. The program reads the input data from the input file *inmat.dat* where the input vectors corresponding to all of the samples of characters are stored. The initial weight matrix is read from the file *wtmat.dat* . Using the conventional backpropagation algorithm the weight matrix is opdated and stored in the file *optwtmat.dat*.

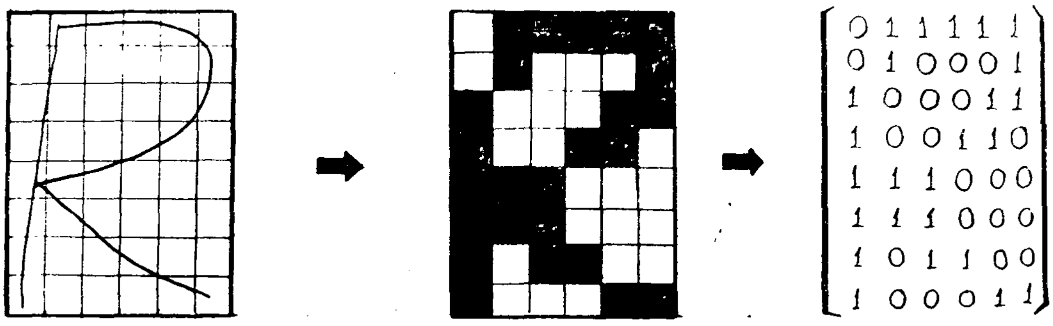
The second section of the program again reads the rest ten training inputs from the same file *inmat.dat* . The input vector is multiplied with the optimal weight matrix obtained from the first section and after applying the squashing function the output vector is obtained. The program calculate the error for the network as discussed earlier. According to the algorithm, if the error decreases, then the size of the network is increased and again the first section is executed. This process is repeated until the error reaches its first minimum and finally it prints the optimal number of layers and number of neurons in each layer. To examine the relationship between the network size and its ability to recognise complex patterns, the number of characters correctly recognised during the second training phase is also reported. For this purpose, the output vector is digitized by taking 0.5 as the threshold level. So any element of the output vector is considered as zero if it is less than or eqaual to 0.5; otherwise it is considered as one. The digitized output vector is compared with the desired output vector. If it matches then it is decided that the network has correctly recognised that particular character. For each network configuration, the number of correctly recognised patterns are determined.

4.3 RESULS AND DISCUSSION

The performance of the network with different size during the second phase of training is given in Table I and Table II. The number of correctly recognised patterns out of given ten input pattrens is shown. The corresponding training times for getting the temporary optimal weight matrix (termed as *initial training time*) and the time required to calculate the error (termed as *testing time* during which it tests whether the network is optimal or not) for each case is also shown.

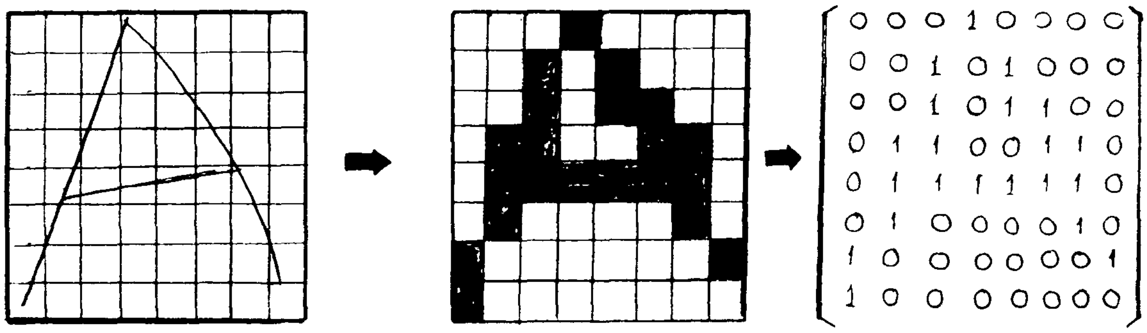
By analysing the results carefully it has been observed that by simply increasing the size of a network does not improve its performance; rather the training time increases. In particular, let us consider the eighth training pattern of the letter R given to the network which is shown in Fig. 4.3 . The network with a single layer could not recognise it. When the number of layers is increased to two it became able to recognise the letter. But when the number of layers is still increased to three it again failed to recognise that particular character. Also, it has been observed that by increasing the number of neurons arbitarily large, does not help the situation. So it is important to increase the number of layers and neurons step by step and stop the process when the performance starts degrading.

In the first case, the network with an input layer having 48 (8X6) neurons is found to be optimal when the number of layers is two and the number of neurons in the hidden layers are 48X48 and 48X6 . This corresponds to the obsevation number 5 of the Table I. In the second case, the network having 64 (8x8) neurons in the input layer is found to be optimal with two hidden layers having 64X64 and 64X6 number of hidden units. This corresponds to the observation number 5 of the Table II.



$$\begin{bmatrix} 0&1&1&1&1&1\\ 0&1&0&0&0&1\\ 1&0&0&0&1&1\\ 1&0&0&1&1&0\\ 1&1&1&0&0&0\\ 1&1&1&0&0&0\\ 1&0&1&1&0&0\\ 1&0&0&0&1&1 \end{bmatrix}$$

FIG.4.1 REPRESENTATION OF A LETTER USING 8X6 BOXES



$$\begin{bmatrix} 0&0&0&1&0&0&0&0\\ 0&0&1&0&1&0&0&0\\ 0&0&1&0&1&1&0&0\\ 0&1&1&0&0&1&1&0\\ 0&1&1&1&1&1&1&0\\ 0&1&0&0&0&0&1&0\\ 1&0&0&0&0&0&0&1\\ 1&0&0&0&0&0&0&0 \end{bmatrix}$$

FIG.4.2 REPRESENTATION OF A LETTER USING 8X8 BOXES

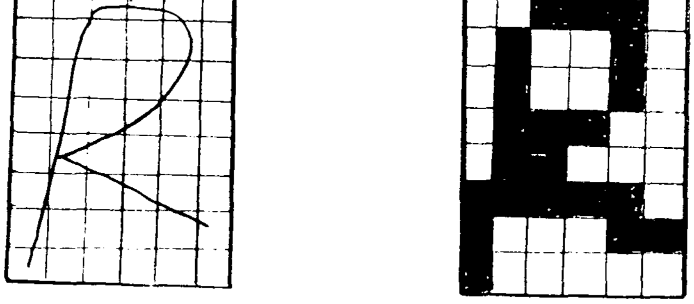


FIG.4.3 THE LETTER FOR WHICH THE NETWORK SHOWS TYPICAL CHARACTERISTIC

OBS. NO	NO. OF LAYERS	NO. OF NEURONS IN INPUT LAYER	NO.OF NEURONS IN 1ST HIDDEN LAYER	NO.OF NEURONS IN 2ND HIDDEN LAYER	NO.OF NEURONS IN 3RD HIDDEN LAYER	NO.OF NEURONS IN THE OUTPUT LAYER	NO.OF CHARACTERS CORRECTLY RECOGNISED (OUT OF 10)	INITIAL TRAINING TIME/CHARACTER (MIN : SEC)	TESTING TIME (SEC)
1.	1	40	40 X 3	-	-	3	7	0 : 25.35	3.11
2.	2	40	40 X 40	40 X 3	-	3	10	1 : 28.34	22.96
3.	3	40	40 X 40	40 X 10	10 X 3	3	9	3 : 07.49	38.67
4.	1	40	40 X 6	-	-	6	9	0 : 33.74	3.77
5.	2	40	40 X 40	40 X 6	-	6	10	2 : 16.43	29.45
6.	3	40	40 X 40	40 X 20	20 X 6	6	10	4 : 47.34	46.96
7.	1	40	40 X 9	-	-	9	8	0 : 38.47	5.09
8.	2	40	40 X 40	40 X 9	-	9	9	3 : 24.46	32.39
9.	3	40	40 X 40	40 X 30	30 X 9	9	10	5 : 46.17	53.04

TABLE I

OBS. NO	NO. OF LAYERS	NO. OF NEURONS IN INPUT LAYER	NO.OF NEURONS IN 1ST HIDDEN LAYER	NO.OF NEURONS IN 2ND HIDDEN LAYER	NO.OF NEURONS IN 3RD HIDDEN LAYER	NO.OF NEURONS IN THE OUTPUT LAYER	NO.OF CHARACT-ERS CORRECTLY RECOGNISED (OUT OF 10)	INITIAL TRAINING TIME/CHARACTER (MIN : SEC)	TESTING TIME (SEC)
1.	1	64	64 X 3	-	-	3	9	0 : 31.23	3.56
2.	2	64	64 X 64	64 X 3	-	3	10	2 : 07.05	37.07
3.	3	64	64 X 64	64 X 10	10 X 3	3	7	4 : 17.44	54.49
4.	1	64	64 X 6	-	-	6	9	0 : 38.53	4.16
5.	2	64	64 X 64	64 X 6	-	6	10	3 : 36.10	57.16
6.	3	64	64 X 64	64 X 20	20 X 6	6	9	6 : 09.17	64.15
7.	1	64	64 X 9	-	-	9	8	0 : 49.02	5.73
8.	2	64	64 X 64	64 X 9	-	9	10	4 : 03.67	61.23
9.	3	64	64 X 64	64 X 30	30 X 9	9	9	6 : 43.29	72.04

TABLE II

CONCLUSIONS

The work demonstrates that by just increasing the size of the network arbitarily large, does not improve the performance of the network; rather the training time increases. A non-static algorithm has been proposed and implemented which decides the optimal size of a network for a particular class of problems. A systematic study of the relationship between the size and structure of the network and the performance has been done and computational results presented. This learning paradigm could form an enabling core for complex problems in nonlinear adaptive control, object recognition and behavioural conditioning. Furthermore, the combination of the proposed non-static algorithm and segregation algorithm will give a new direction to the dynamic system theory of neural networks.

REFERENCES

1. Bart Kosko

   Neural Networks and Fuzzy Systems

2. Branko Soucek & Marina Soucek

   Neural and Massively Parallel Computers

   Published by Weily Interscience Publications

3. D. Hammerstrom

   Neural Network learning

   IEEE Coference on Tools for AI , Nov. 1991

4. Dan Hammerstron, Cris Koutsougeras & Gerald G. Pechanek

   Are Neural Networks a tool for AI ?

   Proc. of 1991 IEEE,

   International Conference on Tools for AI, Nov. 1991

5. David Baily & Donna Thompson

   How to develop neural network applications

   AI Expert, Vol.5, No.6, June 1991

6. F. Schonbaur & M. Kohle

   A non-satic learning paradigm for neural networks

   Artificial Intelligence in Manufacturing

7. Jocelyn Sietsma & Robert J.F. Dow

   Creating artificial neural networks that generalise

   Nerural Networks , Vol. 10,1990

8. Kenneth G. Schweller & Anneta L. Plagman
Neural Networks and alphabets : Introducing students to neural networks
SIGCSE Bulletin, Vol.21, No.3, Sept. 1989

9. Lalit Gupta, Mohammad R. Sayeh & Ravi Tammana
A neural network approach to robost shape classification
Pattern Recognition, Vol.23, No.6, 1990

10. Morris W. Firebaugh
Artificial Intelligence : A Knowledge Based Approach

11. Philip D. Wasserman
Neural Computing : Theory & Practice
Published by Van Nostrand Reinhold, New York.

12. Richard P. Lippman
Pattern classification using neural networks
IEEE Communication Magazine, Nov. 1989

13. Robert Hecht Neilson
Neurocomputing

14. S.Gulati, I. Barhen & S. S. Iyengar,
Neurocomputing formalism for computational learning and machine intelligence
Advances in Computers, Vol.33, 1991

15. Yaser S. Abu-Mostafa
Information Theory,Complexity and Neural Networks
IEEE Communication Magazine, Nov. 1989

16. Yoshio Hirose, Koichi Yamashita & Shimpei Hijiya
Backpropagation which varies the number of hidden units
Neural Networks, Vol.10, 1990