AN ARTIFICIAL NEURAL NETWORK FOR SONAR TARGET DETECTION

THESIS SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE
OF
MASTER OF TECHNOLOGY
IN
DIGITAL ELECTRONICS

BY

S. RAVINDRANATHAN

DEPARTMENT OF ELECTRONICS
COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY
KOCHI - 682 022

OCTOBER 1991

CERTIFICATE

This is to certify that this thesis entitled AN ARTIFICIAL NEURAL NETWORK FOR SONAR TRAGET DETECTION is a record of successful work carried out at NPOL, Kochi-682021 by Mr. S. Ravindranathan during the period from January 1991 to October 1991 in partial fulfilment of the requirements for the award of the degree of Master of Technology in Digital Electronics of Cochin University of Science & Technology, Kochi-682022 and has not been submitted anywhere else for M.Tech Degree.

Dr. K.G. Nair,
Professor and Head,
Dept. of Electronics,
C U S A T
Kochi - 682022.

CERTIFICATE

This is to certify that the thesis entitled **AN ARTIFICIAL NEURAL NETWORK FOR SONAR TARGET DETECTION** submitted by **Shri S. RAVINDRANATHAN** to the **Cochin University of Science and Technology, Cochin,** in partial fulfilment of the requirements of the degree of **Master of Technology** in **Digital Electronics,** is a bonafide record of the work carried out by him under my guidance and supervision.

[illegible]

Dr. A. Unnikrishnan
Scientist E
Naval Physical & Oceanographic Laboratory
Cochin 682 021

Thrikkakara
21st October 1991

My sincere THANKS to

The Director, NPOL for kindly permitting me to carry out this work at NPOL.

Dr. A.Unnikrishnan for his sincere and efficient guidance.

M/s V.Chander, **Abraham Eapen, K.Sathishkumar & V.P.Felix,** Scientists, NPOL for their constant encouragement and enthusiastic support.

M/s C.K.Joseph, **R.Sivakumar,** Vinod **Robert** and **Prasithlal,** Scientists, DRDO - Cochin University Computer Center for their valuable help in the computer simulations connected with this work.

All my colleagues for their hearty co-operation which made this work reach its completion.

CONTENTS

ABSTRACT

Neural Network has emerged as the topic of the day. The spectrum of its application is as wide as from ECG noise filtering to seismic data analysis and from elementary particle detection to electronic music composition. The focal point of the proposed work is an application of a massively parallel connectionist model network for detection of a sonar target. This task is segmented into:

(i) generation of training patterns from sea noise that contains radiated noise of a target, for teaching the network;

(ii) selection of suitable network topology and learning algorithm and

(iii) training of the network and its subsequent testing where the network detects, in unknown patterns applied to it, the presence of the features it has already learned in.

A three-layer perceptron using backpropagation learning is initially subjected to a recursive training with example patterns (derived from sea ambient noise with and without the radiated noise of a target). On every presentation, the error in the output of the network is propagated back and the weights and the bias associated with each neuron in the network are modified in proportion to this error measure. During this iterative process, the network converges and extracts the target features which get encoded into its generalized weights and biases.

In every unknown pattern that the converged network subsequently confronts with, it searches for the features already learned and outputs an indication for their presence or absence. This capability for target detection is exhibited by the response of the network to various test patterns presented to it.

Three network topologies are tried with two variants of backpropagation learning and a grading of the performance of each combination is subsequently made.

CHAPTER 1

INTRODUCTION

The history of mankind is a never-ending chain of human endeavours for survival in the struggle for existance and for achieving mastery over the physical nature.Science and technology which are the outcome of human intelligence, have helped mankind fight a long way through in this war for supremacy. Innumerable are the wonders of nature and inspiring are their ways of manifestations. In man's inquisitive efforts to unravel the mysteries around him, nothing in nature have been spared from his dissection table - not even himself ! Many of our scientific achievements are the results of such observations and the successful efforts to mimic nature. The shape of ships and submarines are strikingly similar to that of the big fishes. Birds floating in the infinite blue sky were the inspiring source behind the development of aircrafts. Had it not been for our knowledge of the optical system in animals, the colourful world of photography would never have become a reality. It is rather astounding to understand that a primitive form of radar has already been used by bats since time immemorial as their navigational aid ! Recent investigations into the structure and functioning of the human brain and the associated phenomena of cognition and perception have added yet another marvel to the above. And that is the marvel of Artificial Neural Networks (ANN).

1.1 Computers Versus Human Brain

In the world of popular science, the modern computer is often referred to as the "electronic brain". Computers, as they are known today, work in an entirely different manner as compared to the human brain. A numerical calculation performed within a split-second by a supercomputer might take centuries for a human to complete. But the computer cannot do some simple tasks which even an infant baby can, like identifying its mother's face among a few unfamiliar faces. Due to these inherent limitations, it will be a long time before the brain can be substantially mimicked.

Computers and human brain differ basically in their mode of approach to problems and in the way they perform. Conventional computers employ the von Neumann architecture, are logical in execution and can only do logical operations well. Though it is rather impossible for a human brain to surpass the phenomenal speed, accuracy and efficiency of a computer, the computers are left far behind by the brain in solving problems relating to machine-vision and speech recognition.

In essence, it is the difference in design that can account for the difference between the two systems. Computers are designed to carry out instructions sequentially and extremely fast, whereas brain works with many more slower units working in a highly parallel fashion. Such a parallel style is most suited for problems of vision or speech recognition which are also highly parallel in nature.

1.2 Neural Networks - A Historical Perspective

As early as 1940, neurobiologists and neuroanatomists had come to understand about the brain's " wiring " - which they called "neural networks" - involving hundreds of billions of neurons, each connecting to hundreds or thousands of others, but little of its operation was known. It was W.S.McCulloch and W.Pitts (1943) who showed how these networks could compute; but however, the question as to how they could learn remained unanswered until Donald Hebb proposed the hypothesis [32] called "Hebbian learning" in 1949. Hebb's proposal, which became the starting point for the development of learning algorithms for artificial neural networks, had to wait till 1951 when Dean Edmonds and Marwin Minsky succeeded in building their learning machine. Although Minsky was perhaps the first to come up with a learning machine, the real onset of meaningful learning in neuron-like networks can be traced to the work of Frank Rosenblatt [33] who invented a class of simple neuron-like learning networks called "perceptrons". The techniques of digital computer simulation and formal mathematical analysis which are of fundamental importance to neural network analysis were pioneered by him.

Early successes produced a burst of activity and optimism and it seemed that the secret of intelligence had been found and that the human brain was as simple as a large enough network ! This illusion was soon dispelled when the networks failed to solve some of the problems (which the brain could very well solve !). This led to intense diagnostic analysis by Minsky, S.Papert and others who developed rigorous theorems regarding network operation (1969).

Though the discouraged researchers left the field for more promising areas, dedicated scientists like Teuvo Kohonen, Stephen Grossberg and James Anderson continued their efforts facing many hardships. The research papers published during the period from 1970 to 1980 set a strong theoritical foundation, upon which the more powerful multilayer networks of today are being constructed.

In the past few years, there has been an explosive increase in the amount of research activity in the field of neural networks resulting in regular international conventions, dedicated journals and special issues of journals on neural networks and a flood of research papers in other publications. The substantial amount of innovative investigations parallelly going on in the field of hardware have already resulted in the introduction of a few neural network chips also in the market [11-23].

1.3 Neural Network for Sonar Signal Processing

Sonar signal processing is intented to achieve

(a) detection of targets like submarines, surface ships, torpedoes etc. and (b) accurate estimation of their range, bearing and speed (Doppler). A sonar system has to fulfil these missions under extremely adverse environments like:

(1) propagation peculiarities due to sound velocity variation with ocean depth

(2) spreading and absorption losses

(3) contaminating noise

(4) spatial coherence of signals

(5) instability of sonar platform at high sea-states

(6) low data rates due to low velocity of acoustic propagation

(7) extremely stringent dynamic range requirements

The sonar signal processors have to take the above important factors into account. The special features of the sonar environment have to be made use of to the best to realise the most efficient processor that gives maximum probability of detection with minimum false alarm. The ocean environment being nonstationary, the processor has to be adaptive so as to adjust itself to the changing scenario.

Traditional pattern recognition techniques are often used to interpret complex sonar signals. To reduce the amount of computation and to achieve accurate classification, often simplifying assumptions are made about the structure of these signals. For applications where such assumptions are valid, these techniques do perform well. However, if the signals are not simply distributed or are highly correlated, these methods may be inadequate and other more general techniques available are often impractical [25].

In this context, the multi-layered neural networks, which are massively parallel in nature, provide potential alternatives to traditional pattern recognition methods. The learning algorithms they use make far restrictive assumptions about the input pattern structure. Their inherent parallelism allows very rapid parallel search and best-match computations. Capabilities for failure tolerance, error correction and self-organization along with optimised system complexity render neural networks excellent tools for sonar and other applications.

The sophisticated nature of sonar signal processing, coupled with the difficulty to use conventional pattern recognition techniques has been the motivation behind the work under discussion , which explores the possibility of using neural networks for sonar target detection and classification. The chapters to follow, therefore, summarizes the efforts to evolve a neural network for this purpose. Chapter 2 introduces the concept of neural networks and evolves the idea of neural computing using Artificial Neural Networks. The technique of backpropagation to enhance the capability and coverage of neural computing is surveyed in Chapter 3. Chapter 4 outlines the problem of Sonar Target Detection, elaborating on the diverse and complex nature of the problem. The architecture and implementation of a neural network for target recognition, proposed by the author, are discussed in Chapter 5. The results of the simulated runs of the neural network are summarized in Chapter 6. A brief survey of the hardware aspects of neural networks is made in Chapter 7. Various prospective applications of neural networks in sonar technology which are worth further investigation are also discussed in this chapter.

CHAPTER 2

NEURAL NETWORKS AND NEURAL COMPUTING

2.1 The Structure of the Brain

The human brain is one of the most complicated system that has been studied in detail, but still vaguely understood. It contains over one hundred billion basic computing elements called "neurons". Each of these neurons is connected to about ten thousand others by information channels (Fig 2.1) and may have many input signals but is limited to one and only one output signal. Those inputs that are not outputs from other neurons are then inputs from the outside world. Though the neuron shares many characteristics with the other cells in the body, it has unique capabilities to receive, process and transmit electrochemical signals over the neural pathways that comprise the brain's communication system. This network of neurons, called Biological Neural Network(BNN), is responsible for such phenomena as thought, emotion and cognition.

2.2 The Biological Neuron

The neuron is the fundamental building block of the brain and is a stand-alone analogue logical processing unit whose inputs and output are related usually by first-order ordinary differential equations.

- 38680 -

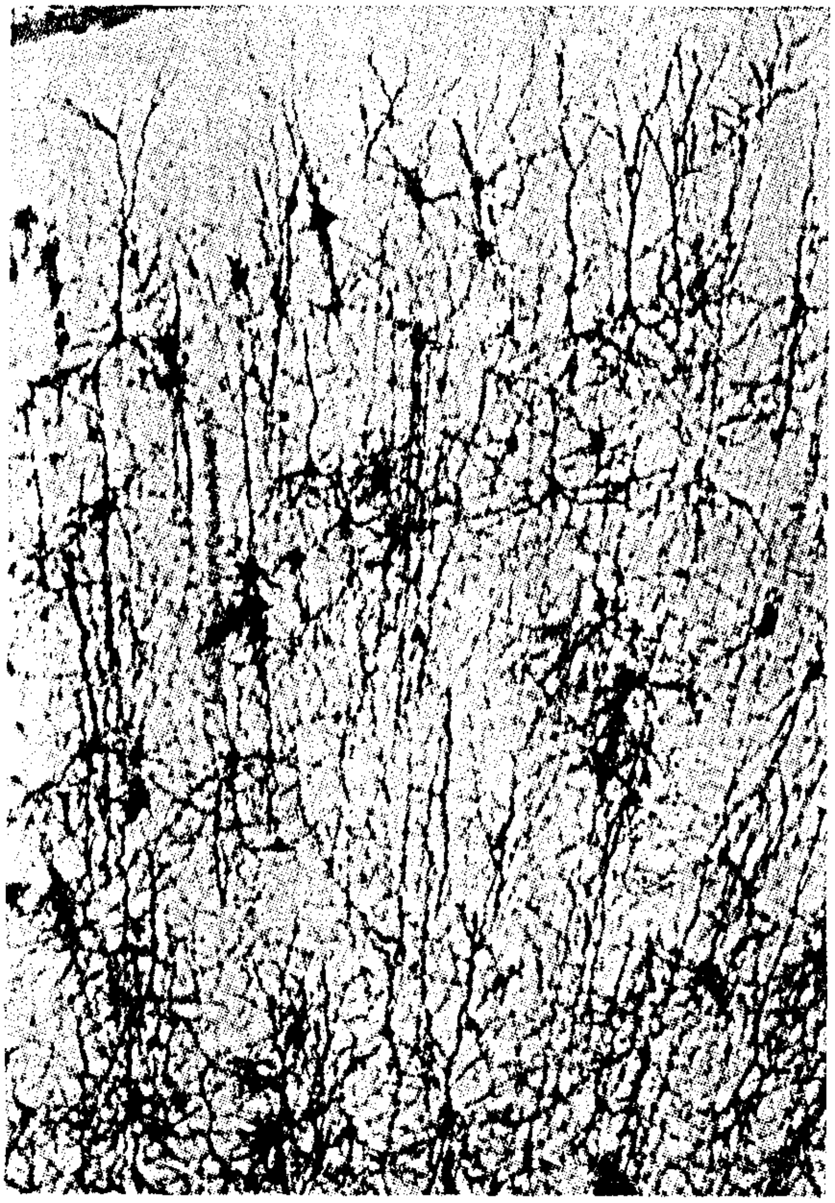


Fig. 2.1 The Biological Neural Network

As shown in Fig 2.2, the neuron consists of three sections : "soma", dendrites and the axon. The soma is the cell body of the neuron. Its outer membrane has the unique capability of generating nerve impulses which is a vital function of the nervous system and central to its computational abilities. Input signals from other neurons enter the soma through the long, irregularly shaped and complexly branched filaments called dendrites. On the dendrites are synaptic connections where signals are received from other neurons (Fig 2.3). The axon serves as the output channel of the neuron. Near its end, the axon has multiple branches, each terminating in a synapse. The axon is a nonlinear threshold device producing a voltage pulse called the "action potential" when the potential within the soma rises above a critical threshold.

The axon of a neuron is coupled with the dendrite of another through a specialised contact called a "synapse". Under the influence of the action potential, the synaptic vesicles release chemicals called "neuro-transmitters" which diffuse across the synaptic cleft and chemically activate gates on the dendrites. These gates, when open, allow the flow of charged ions thereby inducing a voltage pulse on the dendrite and it is conducted along into the next neuron body (Fig 2.4). Since the strength of the induced signal depends on the number of neuro-transmitters emitted, the synapse provides a weighted electrical connection.

Some neuro-transmitters are excitatory and others

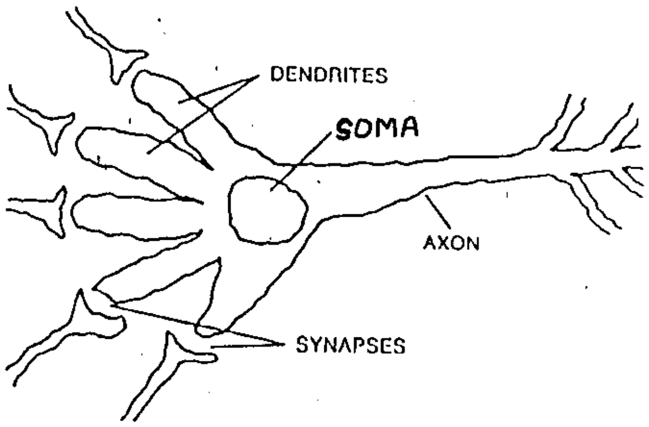


Fig. 2.2 The Biological Neuron

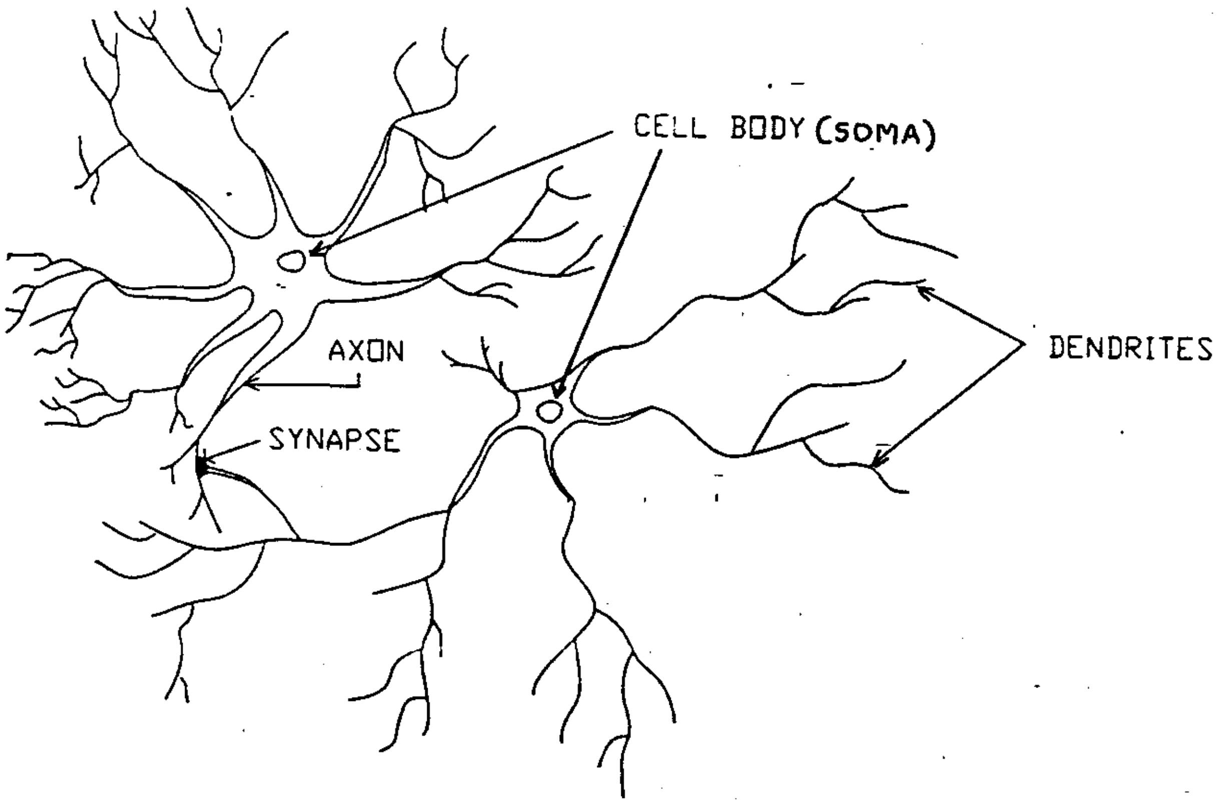


Fig. 2.3 Neuron Interconnection

are inhibitory. The soma combines the signals received over its dendrites and if the resultant signal is above a threshold, it fires and a pulse voltage thereby produced, propagates down the axon to other neurons. Thus, a single neuron can generate a pulse that will activate or inhibit hundreds or thousands of other neurons each of which can, in turn, act upon hundreds or thousands of other neurons. It is this high degree of connectivity rather than its functional complexity that gives the neuron its computational power.

2.3 Learning in Brain

As explained above, the huge computation rate is easily achieved in the brain by employing a massively parallel distributed processing procedure that employs a huge number of simple processing elements viz. the neurons. Learning is thought to occur in brain when modifications are made to the synaptic weights that couple the neurons. It is a process of self-organization and adaptation based on the environmental inputs to the brain from the outside world.

Since many neurons are involved in the brain's computations, the contribution from a single one is not too significant. Thus, the failure of a neuron doesnot affect the performance of others. As the brain learns, it adjusts to this permanent loss of one of its neurons and brings in new ones. This is called "fault tolerance" which is a vital feature of brain's operation. In case of continuing damage, parallel distributed processing systems exhibit a "graceful degradation" where the system performance slowly falls from a high to a reduced level instead of dropping abruptly to zero.

2.4 The Artificial Neuron

Artificial neural net models attempt to achieve real-time response and brain-like performance using many simple processing elements viz. artificial neurons operating in parallel as in BNNs. Since the idea behind neural computing is to produce computing systems having many useful properties of the brain by modelling the major features of the brain and its operation, it is essential that the model functionally resembles the original to the utmost possibility.

The basic features of the simple biological neuron, which were discussed in the previous sections, are depctited in Fig 2.5 . The artificial neuron was designed to mimic the first-order characteristics of the biological neuron viz. (a) control of the electrochemical signals through the dendrites by the synaptic strengths (b) combination of the controlled signals and (c) thresholding of the combined signal. Replacement of the above three features respectively with similar ones as (a) signal multiplication with weights (b) summation of the weighted signals and (c) application of a threshold function to the summed up signal results in a basic model of artificial neuron, shown in Fig 2.6 . A comparison between the BNN and ANN on various features is made in Table 2.1 .

Fig 2.6 depicts the functions associated with an artificial neuron, say the j th one, which is part of an ANN. Inputs $x_1, x_2, \ldots\ldots, x_N$ to this neuron are the outputs from other neurons 1, 2,........., N preceding it and its output y_j (which is also referred to as the activation of unit j)fans out to serve as the inputs to the neurons following it. A network with this type of signal flow is called a feed-forward network. Since the model neuron thresholds the weighted sum of its inputs,

$$\mathrm{NET}_j = \omega_{1j} x_1 + \omega_{2j} x_2 + \ldots\ldots\ldots\ldots + \omega_{Nj} x_N$$

$$= \sum_{i=1}^{N} \omega_{ij} x_i \qquad (2.1)$$

In vector form it can be written as

$$\mathbf{NET} = \mathbf{XW} \qquad (2.2)$$

If f denotes the activation function, then

$$y_j = f(\mathrm{NET}_j) \qquad (2.3)$$

The activation function is generally nonlinear and the type of nonlinearity characterises the behaviour of the neuron. The common types of nonlinearities viz. hard limiters, threshold logic elements, sigmoidal nonlinearities and hyperbolic tangent function which are used to calculate the output state of the neuron are shown in Fig 2.7 [9,26]. The nonlinear activation functions are vital to the expansion of the network's computational capability beyond that of the single - layer network [9]. (In all the implementations, the author has used the sigmoidal nonlinearity for the activation function)

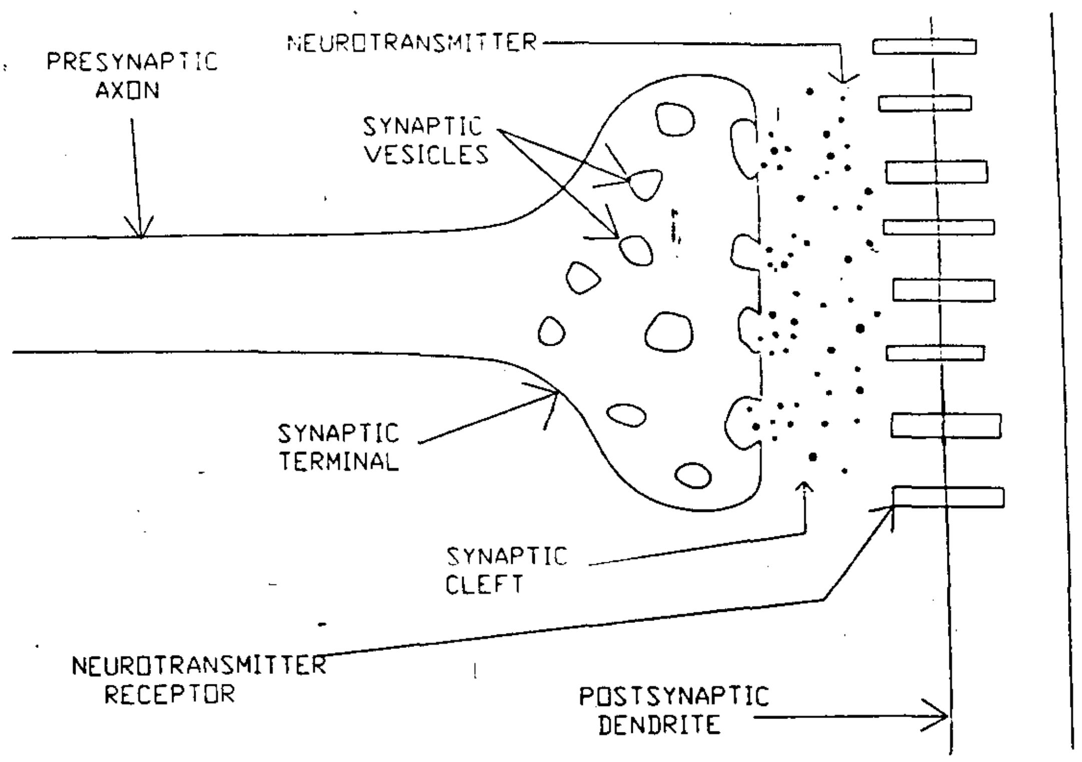


Fig. 2.4   The Synapse

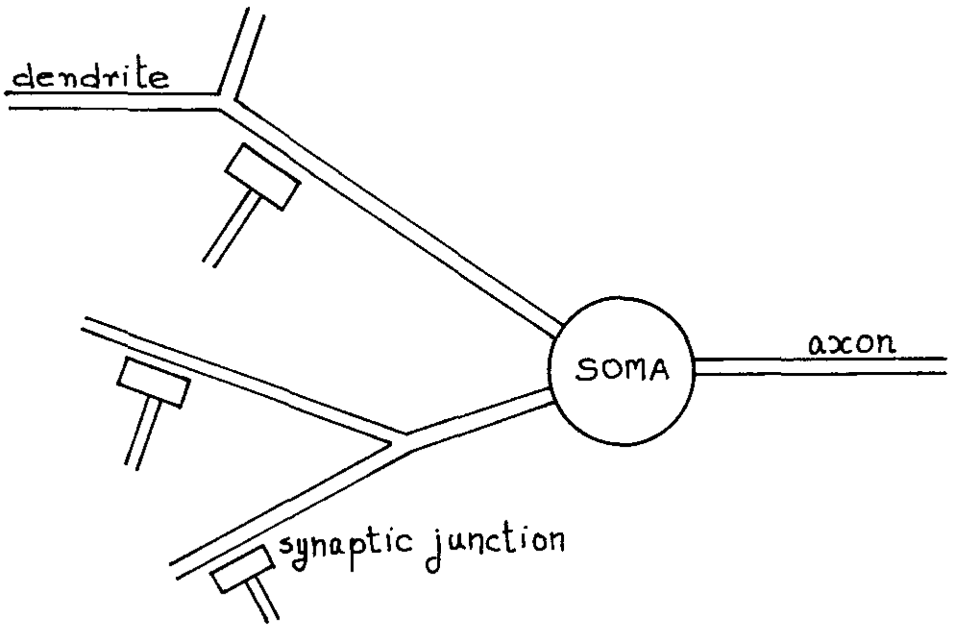


Fig. 2.5   Biological Neuron : Basic Features

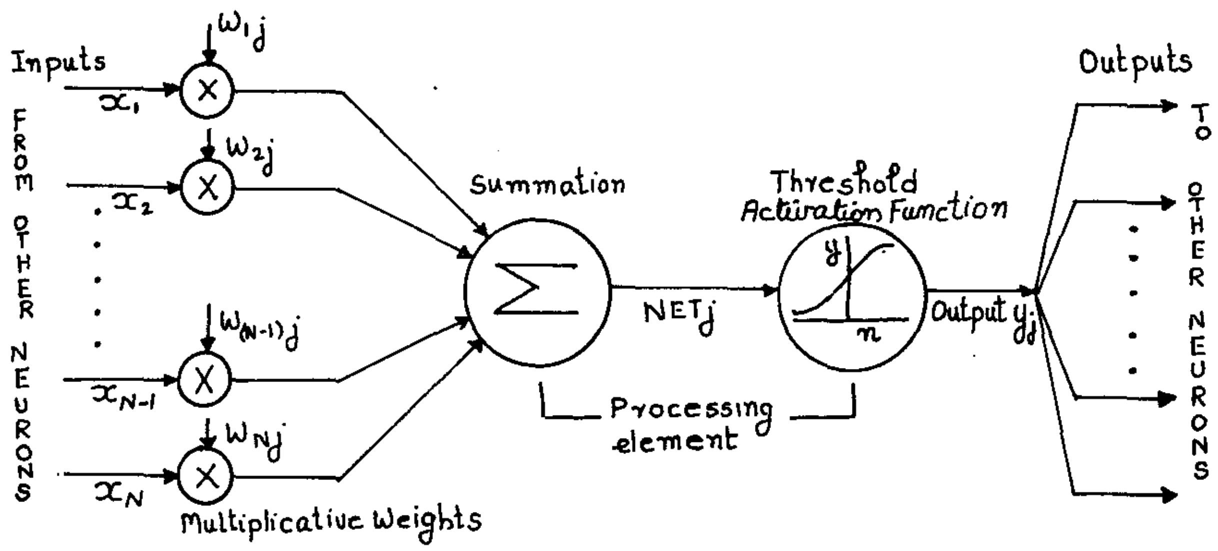
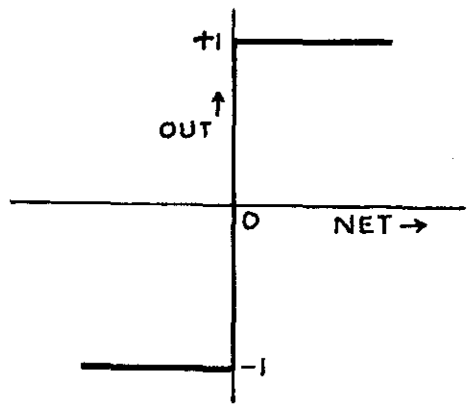
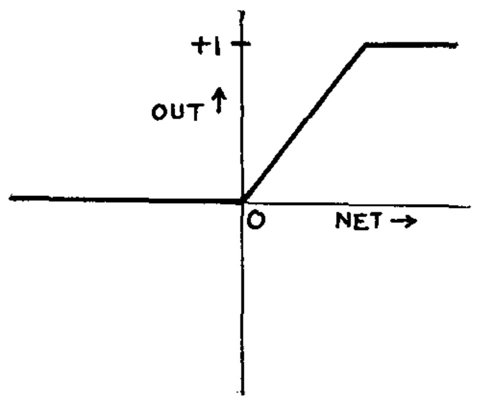


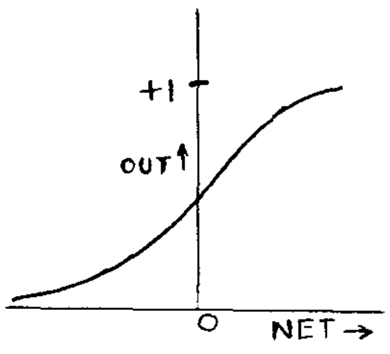
Fig. 2.6 Basic Model of Artificial Neuron



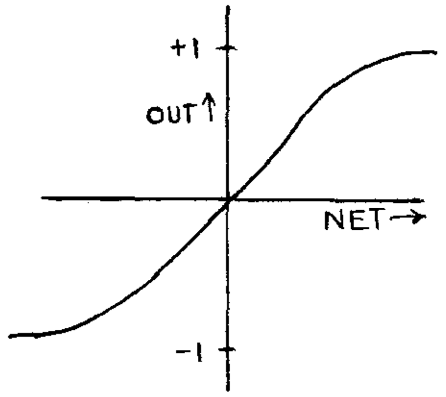
Hard Limiter



Threshold Logic



Sigmoid



Hyperbolic Tangent Function

Fig. 2.7 Neuron Threshold Functions

TABLE 2.1

Comparison Between BNN and ANN

Element	Biological Neural Network	Artificial Neural Network
Organization	Network of neurons	Network of processing elements
Components	Dendrites and axons Synapses Summer Threshold	Inputs and outputs Weights Summation function Threshold function
Processing	Analog	Analog or digital
Architecture	10-100 billion neurons	1-1,000,000 processors
Hardware	Neuron	Switching device
Switching speed	1 millisecond	1 nanosecond to 1 millisecond
Technology	Biological	Silicon Optical Molecular

2.5 Artificial Neural Networks (ANN)

A single neuron can perform certain simple pattern detection functions. But the power of neuro-computing comes from connecting neurons into networks. The simplest in this is a single-layer network containing an array of interconnected neurons as shown in Fig. 2.8 .

Cascading a group of single-layer networks constitutes a multi-layered feed-forward ANN (Fig 2.9). Computational capabilities better than that of single layer are offered by the multilayered ones. The performance of the network is found to improve as the number of hidden units is increased [25]. Recurrent type of networks employing feedback connections are also existent. They exhibit properties of a short-term memory since their output state depends partially on their previous inputs.

ANNs, like their biological counterparts, exhibit the ability to learn and recognize input patterns by adjusting the values of the synaptic weights that interconnect the various nodes. This requires training of the network with example patterns which are sequentially applied as inputs while adjusting the weights according to a predetermined procedure called "learning algorithm". During training, which is an iterative process, the network weights gradually converge to values such that each input vector produces the desired output vector.

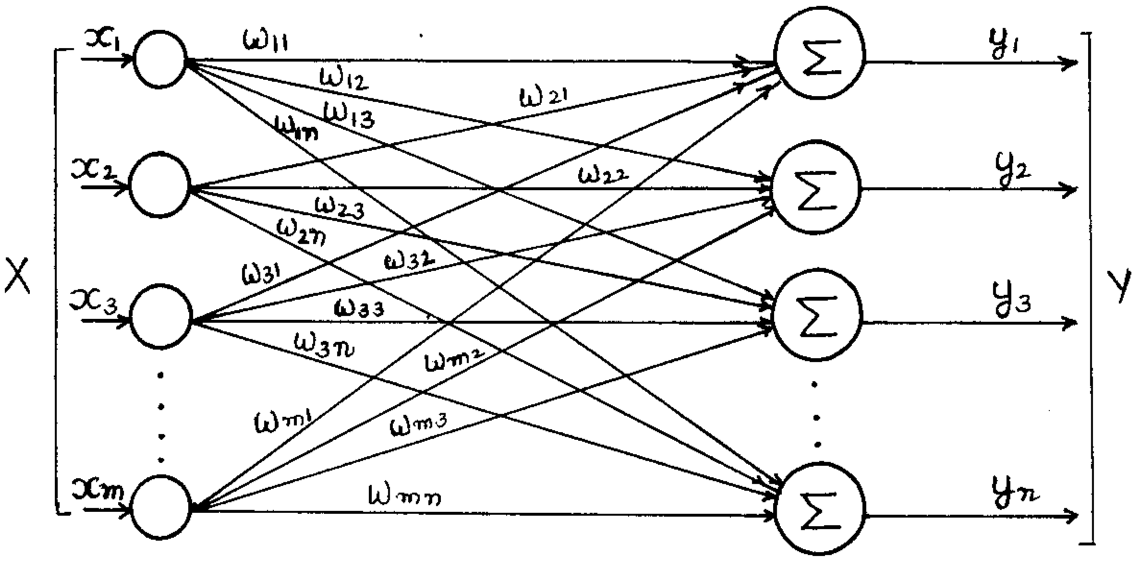


Fig. 2.8 Single-Layer Neural Network

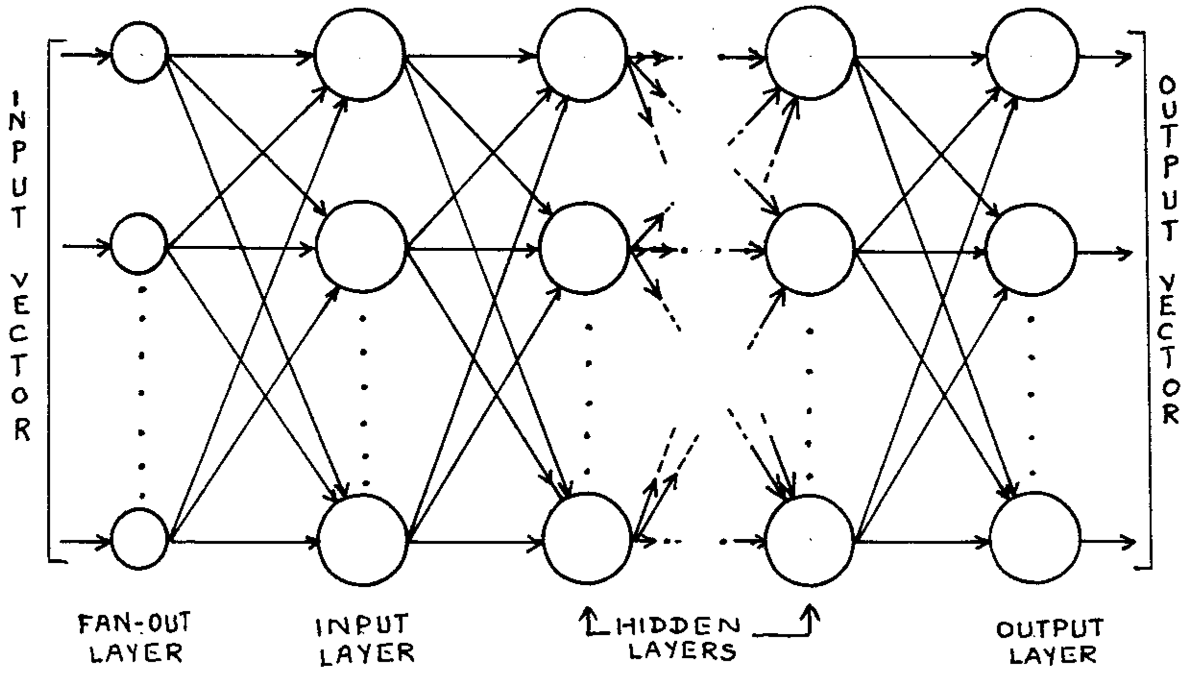


Fig. 2.9 A Multi-Layer Neural Network

The neural network trainings are basically of two types: supervised training and graded (unsupervised) training. In both, it runs through a series of trials. In supervised training, the network is provided with both input vector and the corresponding target vector. After each trial, the network compares its computed output with the target, utilises the difference (error) to modify the weights according to an algorithm that tends to minimise the error; and tries again, iterating until the output error reaches an acceptably low level. In graded training, the network is given input data but no desired output data. Instead, after each trial or series of trials, it is given a grade or performance score that tells it how well it is doing [7,9]. In either case, after training, the network is ready to process and classify genuine inputs.

Neural networks are characterised by (a) the number and modes of synaptic interconnections (b) the node characteristics that are classified by the type of nonlinear elements used and (c) the kind of learning rules implemented. A variety of neural network models which differ in the above features have been evolved for different applications. The most popular among them are listed in Table 2.2 [7].

2.6 Conclusion

The capabilities of earlier feedforward networks were limited and many problems couldnot be solved with such a network. It was then suggested [28] that a multilayer network

TABLE 2.2

Most Popular Neural Networks

Name of network	Inventors and developers	Years introduced	Primary applications	Limitations	Comments
Adaptive resonance theory	Gail Carpenter, Northeastern U.; Stephen Grossberg, Boston U.	1978-86	Pattern recognition, especially when pattern is complicated or unfamiliar to humans (radar or sonar readouts, voiceprints)	Sensitive to translation, distortion, changes in scale	Very sophisticated; not yet applied to many problems
Avalanche	Stephen Grossberg, Boston U.	1967	Continuous-speech recognition; teaching motor commands to robotic arms	Literal playback of motor sequences—no simple way to alter speed or interpolate movements	Class of networks—no single network can do all these tasks
Back propagation	Paul Werbos, Harvard U.; David Parker, Stanford U.; David Rumelhart, Stanford U.	1974-85	Speech synthesis from text; adaptive control of robotic arms; scoring of bank loan applications	Supervised training only—correct input-output examples must be abundant	The most popular network today—works well, simple to learn
Bidirectional associative memory	Bart Kosko, U. of Southern California	1985	Content-addressable associative memory	Low storage density; data must be properly coded	Easiest network to learn—good educational tool; associates fragmented pairs of objects with complete pairs
Boltzmann and Cauchy machines	Jeffrey Hinton, U. of Toronto; Terry Sejnowsky, Johns Hopkins U.; Harold Szu, Naval Research Lab	1985-6	Pattern recognition for images, sonar, radar	Boltzmann machine: long training time. Cauchy machine: generating noise in proper statistical distribution	Simple networks in which noise function is used to find a global minimum
Brain state in a box	James Anderson, Brown U.	1977	Extraction of knowledge from data bases	One-shot decision making—no iterative reasoning	Similar to bidirectional associative memory in completing fragmented inputs
Cerebellatron	David Mar, MIT; James Albus, NBS; Andres Pellionez, NYU	1969-82	Controlling motor action of robotic arms	Requires complicated control input	Similar to avalanche network; can blend several command sequences with different weights to interpolate motions smoothly as needed
Counterpropagation	Robert Hecht-Nielsen, Hecht-Nielsen Neurocomputer Corp.	1986	Image compression; statistical analysis; loan application scoring	Large number of processing elements and connections required for high accuracy for any size of problem	Functions as a self-programming look-up table; similar to back propagation only simpler, although also less powerful
Hopfield	John Hopfield, California Inst. of Technology and AT&T Bell Labs	1982	Retrieval of complete data or images from fragments	Does not learn—weights must be set in advance	Can be implemented on a large scale
Madaline	Bernard Widrow, Stanford U.	1960-62	Adaptive nulling of radar jammers; adaptive modems; adaptive equalizers (echo cancellers) in telephone lines	Assumes a linear relationship between input and output	Acronym stands for multiple adaptive linear elements; powerful learning law; in commercial use for more than 20 years
Neocognitron	Kunihiko Fukushima, NHK Labs	1978-84	Handprinted-character recognition	Requires unusually large number of processing elements and connections	Most complicated network ever developed; insensitive to differences in scale, translation, rotation; able to identify complex characters (such as Chinese)
Perceptron	Frank Rosenblatt, Cornell U.	1957	Typed-character recognition	Cannot recognize complex characters (such as Chinese); sensitive to difference in scale, translation, distortion	The oldest neural network known; was built in hardware; rarely used today
Self-organizing map	Teuvo Kohonen, Helsinki U. of Technology	1980	Maps one geometrical region (such as a rectangular grid) onto another (such as an aircraft)	Requires extensive training	More effective than many algorithmic techniques for numerical aerodynamic flow calculations

(Courtesy, IEEE Spectrum, March 1988)

with backpropagation to adjust the weights can solve a larger class of problems. The chapter to follow discusses the backpropagation algorithm used in multilayer ANNs.

CHAPTER 3

THE BACKPROPAGATION ALGORITHM

The classification capability of the neural network improves with more number of hidden layers and with all layers adaptive. It is a simple matter to adapt the neurons in the output layer, since the desired responses for the entire network are the desired responses for the corresponding output neurons. Given the desired responses, adaptation of the output layer can be a straight forward exercise of the LMS algorithm. But, fundamental difficulty lies in obtaining the desired responses for the neurons in the hidden layers. The *backpropagation algorithm* is a method for establishing desired responses for such neurons. This algorithm was reported earliest by P. Werbos [38], then discovered by D.B. Parker [39] and rediscovered by D.E. Rumelhart, J.L. McClelland and others [2].

3.1 The Perceptron

Network paradigms for pattern recognition were explored by McCulloch and Pitts in their studies on ANNs during the 1940s [34]. The neuron model they proposed is shown in Fig 3.1 . The Σ unit multiplies each input x by a weight w and sums up the weighted inputs. If this sum is greater than a threshold, the output is "*one*" ; otherwise it is "*zero*". These systems and their variants are collectively called "*perceptrons*".

In general, they consist of a single layer of neurons connected by weights to a set of inputs as depicted in Fig 3.2 .

3.1.1 The Perceptron Training

The perceptron learning is a variant of the Hebbian learning proposed in 1949 by Donald Hebb [32]. According to the perceptron learning model, the synaptic strength interconnecting two neurons in a network is increased if both the source and destination neuron are activated. In this way, the often-used paths in the network are strengthened.

A perceptron is trained by presenting a set of patterns to its input, one at a time, and adjusting the weights until the desired output occurs for each of them. A pattern vector **X** is applied to the network input and the output vector **Y** is calculated for the present weight vector **W** from the relation **Y** = **X W** . If **Y** is different from the target value, the weights connecting to inputs enhancing this erroneous result are modified in value to minimise the error. If **Y** is correct, nothing is changed. This process is repeated for all other pattern vectors so that the network generalises to classify them correctly.

The single-layer perceptron, which employs a hard-limiting threshold function, suffers from the "*credit assignment*" problem and hence is seriously limited in its representational ability. Further, the limitations imposed by

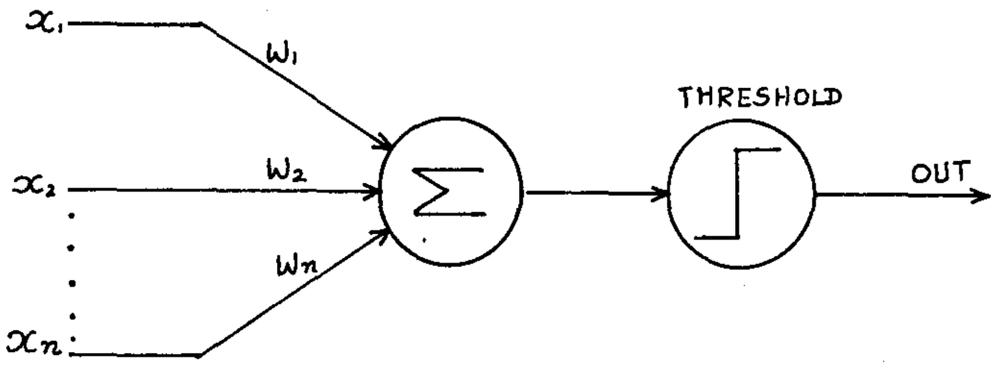


Fig. 3.1 Perceptron Neuron

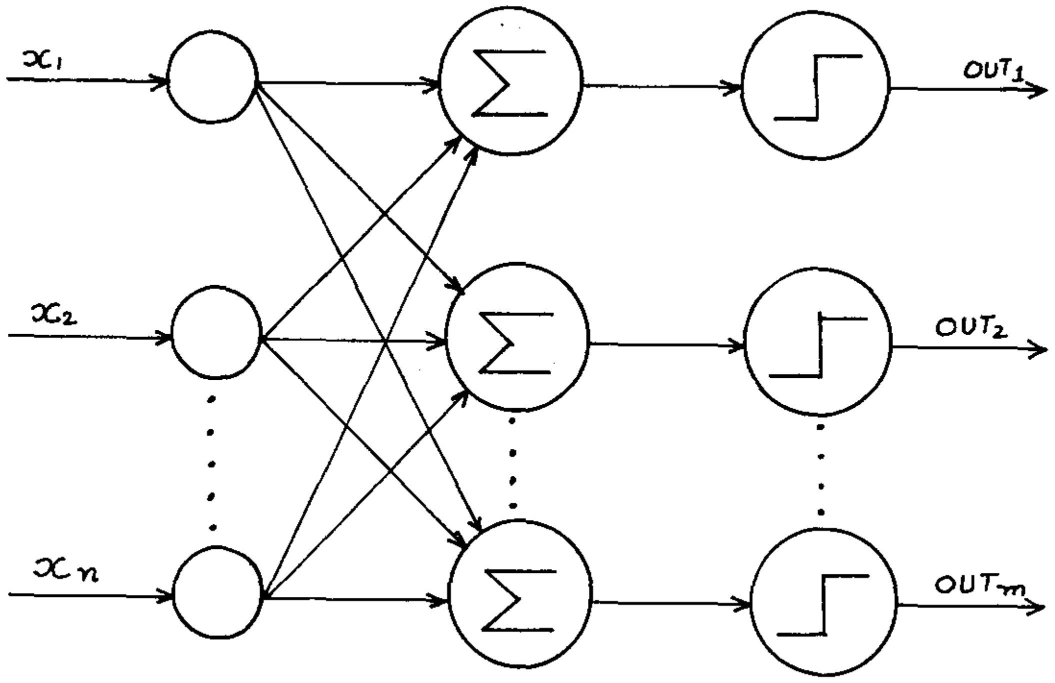


Fig. 3.2 Single-Layer Multioutput Perceptron

linear separability restricts the applicability of single-layer networks only to classification problems in which the components of the input vectors can be separated geometrically with a straight line. A large class of linearly inseparable problems (ex: the Exclusive-Or problem) do set definite bounds on the network capabilities.

Thus, it is important to know beforehand if a given function is linearly separable. But there is no simple way to ensure the linear separability when the number of variables is large. The probability of any randomly selected function being linearly separable become vanishingly small with even a modest number of variables [9]. Also, in many real-world situations, the inputs are often time-varying and may be separable at one time and not at another. Hence single-layer perceptrons are limited to simple problems.

3.2 The LMS Algorithm

One problem with the perceptron convergence procedure is that decision boundaries may ocsillate continuously when inputs are not separable and distributions overlap. A modification to the perceptron convergence procedure can form the least mean square (LMS) solution in this case. This solution minimises the mean square error between the desired output and the actual output of the net. The algorithm that forms the LMS solution is called the Widrow-Hoff or LMS algorithm [35,36].

The LMS algorithm is identical to the perceptron convergence procedure except that the hard-limiting nonlinearity is made linear. Weights are corrected on every trial by an amount that depends on the difference between the desired and the actual outputs. The error function for a given set of weights is computed as the squared error summed over all input patterns and output neurons.

To find a set of weights which minimises the error function, the LMS procedure finds the values of all the weights that minimise this function using the "*gradient descent*" method. Here, after the presentation of each pattern, the error on that pattern is computed and each weight is moved down the error surface gradient towards its minimum value for that pattern. The system thus moves downhill in the weight-space until it reaches the minimum error value. With all the weights having thus reached their minimum, the system has reached equilibrium.

3.3 The Multi-layer Perceptron

When pattern classes cannot be separated by a hyperplane, a network with a more complex structure than the single-layer perceptron is required. Such a feed-forward network, called the *multi-layer perceptron*, has one or more additional hidden layers between the input and the output layer. The neurons in all the layers are similar to that in a perceptron, except that instead of the hard-limiting function, the sigmoid function is used for thresholding. The capabilities of the multi-layer perceptrons to overcome many limitations of their single-layer counterpart stem from this sigmoidal nonlinearity.

The training algorithm for multi-layer perceptron is called the "*generalised delta rule*" or the "*backpropagation rule*" proposed by Rumelhart, McClelland and Williams in 1986 [2]. This algorithm, which is a generalisation of the LMS algorithm, uses a gradient search to minimise a cost function equal to the mean square difference between the desired and the actual net outputs.

Here, the net is trained with supervision. Weights and node biases are initialised to small random values and all training patterns are then presented repeatedly. Weights are adjusted after every trial until the cost function is reduced to an acceptable value or remains unchanged. An essential component of the algorithm is the iterative procedure that propagates error terms back from nodes in the output layer to those in lower layers.

3.4 The Backpropagation (BP) Algorithm

Let

E_p - the output error function corresponding to pattern p.

E - the total of the error function for all the patterns.

t_{pj} - the target (desired) output at node j when the p th pattern is presented at the input.

O_{pj} - the actual output at node j corresponding to p th pattern presented at the input.

W_{ij} - the weight of the connection from node i at the input layer to node j at the output layer.

Since the network is of multi-layered nonrecurrent feed-forward configuration, a node (neuron) in any layer sends its output only to nodes in heigher layers and receives inputs only from nodes in lower layers.

The net output of each unit j, for the pattern p, can be written as

$$net_{pj} = \sum_{i} W_{ij} O_{pi} \qquad (3.1)$$

The output O_{pj} from each unit j is the result of a threshold activation function f acting on the above weighted sum.

$$\text{ie. } O_{pj} = f(net_{pj}) \qquad (3.2)$$

Although any continuously differentiable monotonic function can be used for the thresholding, for the multi-layer perceptron it is the sigmoid function that is used. Its continuity and nonlinearity provide the neurons with the required differentiability along the signal paths of the network and computational power in multi-layer mode [24]. The sigmoid function is defined as :

$$f(net) = \frac{1}{1 + e^{-a\,net}} \qquad (3.3)$$

and has the range $0 < f(net) < 1$. Here 'a' is a +ve constant that controls the "*spread*" of the function. It also acts as an automatic gain control since for small input signals, its slope is steeper, resulting in rapid changes in the function and hence a large gain. For large inputs, the slope and hence the gain is much less. The sigmoid function thus renders the network capable of accepting large inputs without loosing sensitivity to small changes in the signal.

Another advantage of the sigmoidal nonlinearity is that it has a simple derivative which simplifies the implementation of the BP system.

From (3.2) and (3.3),

$$O_{pj} = \frac{1}{1 + e^{-a\,net_{pj}}} \qquad (3.4)$$

so that

$$f'(net_{pj}) = \frac{\partial O_{pj}}{\partial net_{pj}}$$

$$= a\, O_{pj}\,(1 - O_{pj}) \qquad (3.5)$$

Thus, the derivative is a simple function of the outputs. Now, let the error function be proportional to the square of the difference between the actual and desired output for all patterns to be learned. Thus,

$$E_p = \frac{1}{2} \sum_j (t_{pj} - O_{pj})^2 \qquad (3.6)$$

where j ranges over all the output nodes. The factor 1/2 makes the analysis simpler and brings the error function in line with other similar measures.

Considering the error function for all the patterns,

$$E = \sum_p E_p$$
$$= \frac{1}{2} \sum_p \sum_j (t_{pj} - O_{pj})^2 \qquad (3.7)$$

with p ranging over the entire set of input patterns. The objective is to find a set of weights so as to minimise E. The setup being deterministic, the LMS procedure employing the "*gradient descent*" algorithm in the weight-space can be used to arrive at the values of these weights. Thus it is required to compute

$$\frac{\partial E}{\partial W_{ij}} = \sum_p \frac{\partial E_p}{\partial W_{ij}} \qquad (3.8)$$

and move W_{ij} by an amount proportional to the gradient. Since the training patterns are presented to the network in succession during the learning phase, the task is to compute derivatives of E_p with respect to different weights.

Using the chain rule,

$$\frac{\partial E_p}{\partial W_{ij}} = \frac{\partial E_p}{\partial net_{pj}} \frac{\partial net_{pj}}{\partial W_{ij}} \qquad (3.9)$$

But, using (3.1), the second term in the right hand side of (3.9) can be written as

$$\frac{\partial net_{pj}}{\partial W_{ij}} = \frac{\partial}{\partial W_{ij}} \sum_l W_{lj} O_{pl}$$
$$= \sum_l \frac{\partial W_{lj}}{\partial W_{ij}} O_{pl}$$
$$= O_{pi} \qquad (3.10)$$

since $\frac{\partial W_{lj}}{\partial W_{ij}} = \begin{cases} 1, & \text{for } l = i \\ 0 & \text{otherwise.} \end{cases}$

If the change in error can be defined as a function of the change in the net input to a unit as,

$$-\frac{\partial E_p}{\partial net_{pj}} = \delta_{pj} \qquad (3.11)$$

Then (3.9) can be written as

$$-\frac{\partial E_p}{\partial W_{ij}} = \delta_{pj}\, O_{pi} \qquad (3.12)$$

Decreasing the value of E_p thus means making the weight changes proportional to $\delta_{pj}\, O_{pi}$.

ie. $\Delta_p W_{ij} = \eta\, \delta_{pj}\, O_{pi}$ (3.13)

Where $\Delta_p W_{ij}$ is the change made in W_{ij} corresponding to pattern p and η is a step size parameter for the gradient descent (called "*learning rate coefficient*"). A knowledge of the δ_{pj} value for each of the units will help in reducing E.

Applying the chain rule to (3.11),

$$\delta_{pj} = -\frac{\partial E_p}{\partial net_{pj}} = -\frac{\partial E_p}{\partial O_{pj}} \frac{\partial O_{pj}}{\partial net_{pj}} \qquad (3.14)$$

But, from (3.5)

$$\frac{\partial O_{pj}}{\partial net_{pj}} = f'(net_{pj})$$

The value of the derivative of E_p for the outputunits can be obtained from (3.6) as

$$\frac{\partial E_p}{\partial O_{pj}} = -(t_{pj} - O_{pj})$$

Hence for all output units, from (3.14)

$$\delta_{pj} = (t_{pj} - O_{pj})\, f'(net_{pj}) \qquad (3.15)$$

For the output neurons, the values of the target t_{pj} are readily available and the corresponding output O_{pj} is computable using the present weight values. Hence using (3.15) and (3.13), all weights connecting to output units can be updated. But, this is not possible for the hidden neurons since their targets are not known. Thus, if unit j is not an output neuron, using the chain rule of differentiation again,

$$\frac{\partial E_p}{\partial O_{pj}} = \sum_k \frac{\partial E_p}{\partial net_{pk}} \frac{\partial net_{pk}}{\partial O_{pj}}$$

$$= \sum_k \frac{\partial E_p}{\partial net_{pk}} \frac{\partial}{\partial O_{pj}} \sum_i W_{ik} O_{pi}$$

Using (3.1) and (3.11) and noting that the sum drops out since the partial differential is nonzero for only one value as in (3.10),

$$\frac{\partial E_p}{\partial O_{pj}} = \sum_k \frac{\partial E_p}{\partial net_{pk}} W_{jk}$$

$$= \quad - \sum_{k} \delta_{pk} W_{jk} \qquad (3.16)$$

Using (3.16) and (3.5) in (3.14), the error function for a unit j which is not an output unit is given by

$$\delta_{pj} \quad = \quad f'(net_{pj}) \sum_{k} \delta_{pk} W_{jk} \qquad (3.17)$$

If neuron j is in a layer directly below the output layer, then W_{jk} is nonzero only if k is an output unit. If k is an output unit, then its δ_{pk} is known from (3.15) and the modified weights pertaining to neurons in the output layer is computed from (3.13). The units in layer below can now use (3.17) and (3.13) to modify the weights pertaining to them. This process can be carried out till the input layer.

A close observation of the above relations reveals a forward sweep in which outputs of neurons in each layer undergoes a weighted combination to produce inputs to neurons in the next higher layer. But in the learning phase, there is a reverse sweep where errors are propagated backward using the same weights. Networks employing the passing back of the error function during their learning phase are referred to as "*backpropagation networks*". For any pattern, using (3.5), (3.13), (3.15) and (3.17), modified weights for the output layer and the hidden layers are respectively given by

$$W_{ij}(n+1) = W_{ij}(n) + a\,\eta\, O_i\, O_j\, (1 - O_j)(t_j - O_j) \qquad (3.18)$$

and

$$W_{ij}(n+1) = W_{ij}(n) + a\,\eta\, O_i\, O_j\, (1 - O_j) \sum_{k} \delta_k W_{jk} \qquad (3.19)$$

Using the above, weights are progressively modified from the output layer to the input layer side.

3.4.1 Effect of Bias

More rapid convergence of the training process can be achieved by providing each neuron with a trainable bias that offsets the origin of the threshold function. This feature is incorporated into the learning algorithm by adding to each neuron, a weight connected to +1. This weight is trainable in the same way as all the other weights except that the source is always +1 instead of being the output of a neuron in a previous layer.

3.4.2 Shape of the Error Surface

The surface formed by the contour of the error measure in a multi-dimensional "*weight space*" is called the "*error surface*". In networks with hidden neurons, the error surface can be complex and may contain many local minima. Hence it is possible that the steepest descent in weight space may get stuck at any one of the local minima thereby failing to reach the global minimum (corresponding to minimum-most error). If the number of neurons and connections in the network are more than those reuired for the task, poor local minima are rarely encountered [3].

3.4.3 Number of Layers and Neurons

Though more number of hidden layers leads to more complex decision regions in the pattern space and hence to improved power of classification, it is not limitless. In general, a network with three layers of perceptron units can form arbitrarily complex decision regions capable of separating any meshed class of input patterns. The "*Kolmogorov representation theorem*" [37], which demonstrates that a three-layer perceptron can form any continuous nonlinear function of the inputs, states that more than three layers are never needed in a network. Hence, a three-layer perceptron with two hidden layers has surprising computational power and can emulate any traditional deterministic classifier [6,26,28].

The number of neurons in a network must be large enough to have a compatible depth of complexity for the decision region as demanded by a given problem. It must, however, be small enough to enable reliable estimation of the many weights from the available training data. Lippmann [28] states that in a three-layer network, the number of neurons in the second layer should typically be more than three times that in the first layer.

3.4.4 The Momentum Term

Network learning is accomplished by following the energy function down in the steepest direction until it reaches the bottom of a well in the energy landscape. Hence, once entrapped in a local minimum, there is no direction to move in order to reduce the energy further. This renders climbing up the nergy wall and settling at the global minimum rather difficult.

One way of circumventing this is by giving the weight changes some "*momentum*" which introduces into the weight adaptation equation in the BP algorithm an extra term that is proportional to the amount of the previous weight change (Rumelhart, Hinton and Williams - 1986). It produces a large change in the weight if the changes are currently large and will decrease as the changes become less. This means that the network is less likely to get stuck in local minima early on, since the momentum term will push the changes over local increases in the energy function, thus following an overall downward trend. Once a weight adjustment is made, it is '*remembered*' and serves to modify all subsequent weight adjustments.The modified equations for weight adjustment are:

$$W_{ij}(n+1) = W_{ij}(n) + \Delta W_{ij}(n) \quad \text{where}$$
$$\Delta W_{ij}(n) = \eta\, \delta_j\, O_i + \alpha\, \Delta W_{ij}(n-1) \qquad (3.20)$$

The momentum coefficient α is usually set to around 0.9 . For faster learning, both η and α should be high.

3.4.5 Exponential Smoothing

A method of weight modification based on exponential smoothing was proposed by Sejnowski and Rosenberg (1987). It modifies the weight as.

$$W_{ij}(n+1) = W_{ij}(n) + \eta \, \Delta W_{ij}(n) \quad \text{where}$$
$$\Delta W_{ij}(n) = \beta \, \Delta W_{ij}(n-1) + (1-\beta)\, \delta_j \, O_i \qquad (3.21)$$

The smoothing coefficient ß is in the range 0 to 1 . If ß = 0, smoothing is minimum and the entire weight adjustment comes from the newly calculated change. If ß = 1, the new adjustment is ignored and the previous one is repeated. There is a region between 0 and 1 where the weight adjustment is smoothed by an amount proportional to ß. The training rate coefficient η serves to adjust the size of the average weight change.

3.4.6 The BP Training Procedure

The BP training is an iterative process which aims at minimising the mean square error between the actual and the desired output of the network being trained. The various steps involved in the training are:

Step 1 ... Initialise all neuron weights and biases to small random values.

Step 2 ... Present the training pattern (inputs and desired outputs) to the network.

Step 3 ... Calculate the actual outputs of the network.

Step 4 ... Compute the error (ie. difference between the actual and the desired output).

Step 5 ... Adapt weights using the recursive algorithm. Start from the output layer and work backwards through the hidden layers upto the input layer.

Step 6 ... Repeat steps 2 through 5 till the error falls to an acceptably low value.

3.5 Conclusion

Some aspects of the conventional BP algorithm as well as two of its improved versions viz. the momentum method and the exponential smoothing method were considered. Having established the requirement to use multilayer neural networks with backpropagation to solve complex problems, the next chapter analyses Sonar Detection problems, thereby evolving the requirement of a multilayer network with backpropagation for sonar target detection.

CHAPTER 4

SONAR TARGET DETECTION

4.1 Introduction

Underwater warfare today constitutes one of the greatest threats to the freedom of the seas. Modern warships are reasonably well protected against aerial threats by the cover of sophisticated radar systems and long-range lethal weapons. But they are quite vulnerable to underwater threats from submarines because detection of underwater targets is a relatively difficult task. In such a context, the sonar offers a powerful tool for submarine detection. In the present state of the art, a sonar is defined as the method or equipment that employs underwater sound for detecting, locating and identifying objects in the sea. This includes all applications of underwater sound.

4.2 The Sonar Environment

Sonar systems can operate either in "active" mode or in "passive" mode. In the former, a well-defined signal called a "ping" is transmitted into the water medium and the portion of it reflected back from the target, called the "echo", is detected and processed. In passive mode, no intentional transmission is involved. The target is detected by the noise it inadvertently radiates. The process of detection consists of distinguishing the target-radiated noise signature from the ambient noise signature which is already known.

The active and passive modes have their operational advanteges and disadvanteges and the choice among the two is determined purely by the factor of best suitability in a given scenario. But, unless the situation demands otherwise, it is the passive operation that is widely preferred due to zero risk of self-disclosure.

The performance of a sonar is highly influenced by the characteristics of the water medium, sonar equipment and the target and the constraints they impose on the dynamics of sonar operation. They can be listed as :

(a) low data-rate due to low velocity of sound propagation
(b) relative motion between the target and the scatterers
(c) target being extended rather than a point source
(d) spatial distribution of scatterers
(e) coherent relation of the scattered returns to the transmitted signal
(f) strong back-scattering due to medium heterogeneity (reverberation)
(g) acoustic energy losses due to spreading and absorption in the medium
(h) complicated ray-paths and shadow-zones due to sound velocity variations at different water layers
(i) characteristics of the expected target
(j) pollution of target signal by noise

(k) sonar platform instabilities at high sea-states

(l) stringent dynamic-range requirements

(m) tactical aspects

(n) engineering considerations

All these factors render sonar detection a considerably difficult task. The design of an optimum sonar for a given platform should take into account the effects of the above constraints on the sonar parameters which have to be manipulated to achieve the best results.

4.3 Sonar Signal and Noise

The acoustic field that confronts the sonar comprises the desired portion called the "signal" (an echo or radiated noise from a target) and the undesired portion called the "background" (reverberation , self-noise or ambient noise). The sonar equations are no more than a statement of equality between these two portions.

4.3.1 The Echo

The echo pertains to an active sonar transmission. It refers to the useful portion of the transmitted energy that is reflected back to the source by the target. The echo intensity depends on the "target strength" which is an aggregate of the size, geometry, aspect and surface reflectivity of the target, the transmission pulse-width etc. As the reflecting object imparts its own characteristics to the echo, it is used for target detection and classification.

4.3.2 Radiated Noise

Ships, submarines and torpedoes are excellent sources of underwater sound. They require numerous rotational and reciprocating machinery components for their propulsion, control and habitability. This machinery generates vibration which, after transmission through the hull and the sea, appears as underwater sound at a distant hydrophone. The various sources of sonar noise and their interrelationships are illustrated in Fig 4.1 . They are categorised as self-noise and radiated noise. While the former adversely interferes with own-sonar operation, the latter can serve as a lethal weapon by being picked up by an enemy's sonar.

The sources of radiated noise on ships, submarines and torpedoes can be grouped into three major classes as listed in Table 4.1 . A diagramatic view of the sources of machinery noise aboard a diesel-electric vessel is shown in Fig 4.2 . Machinery noise originates inside the vessel as mechanical vibration from the diverse running machines and reaches the water by various processes of transmission and conduction through the hull. Propeller noise originates outside the hull due to the propeller action and by virtue of the vessel's movement through the water. The main source of propeller noise is cavitation induced by the rotating propellers. Hydrodynamic noise is caused by the irregular and fluctuating

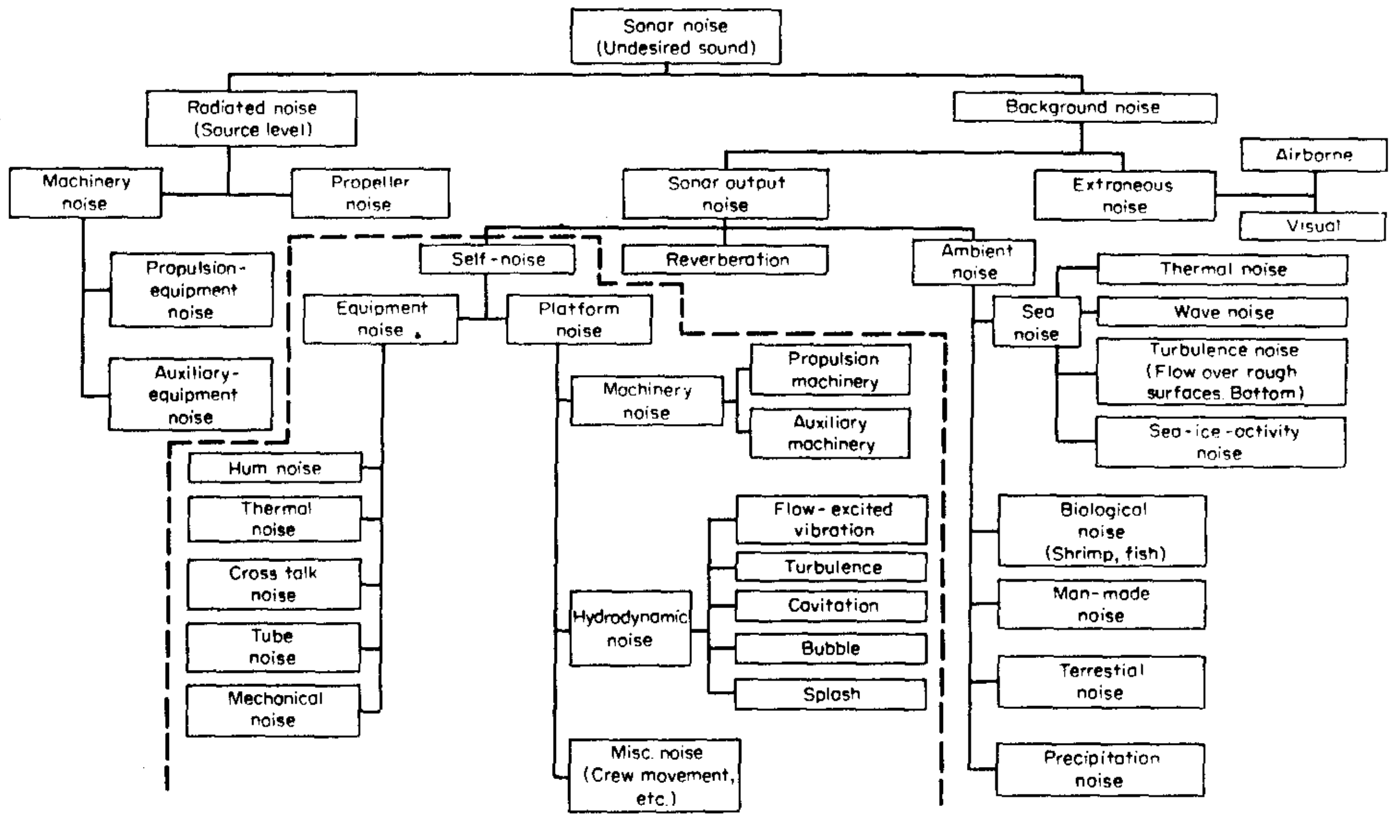


Fig. 4.1 Various sources of sonar noise

TABLE 4.1

Source of Radiated Noise
(Diesel-Electric Propulsion)

Machinery noise:

- Propulsion machinery (diesel engines, main motors, reduction gears)
- Auxiliary machinery (generators, pumps, air-conditioning equipment)

Propeller noise:

- Cavitation at or near the propeller
- Propeller-induced resonant hull excitation

Hydrodynamic noise:

- Radiated flow noise
- Resonant excitation of cavities, plates and appendages
- Cavitation at struts and appendages

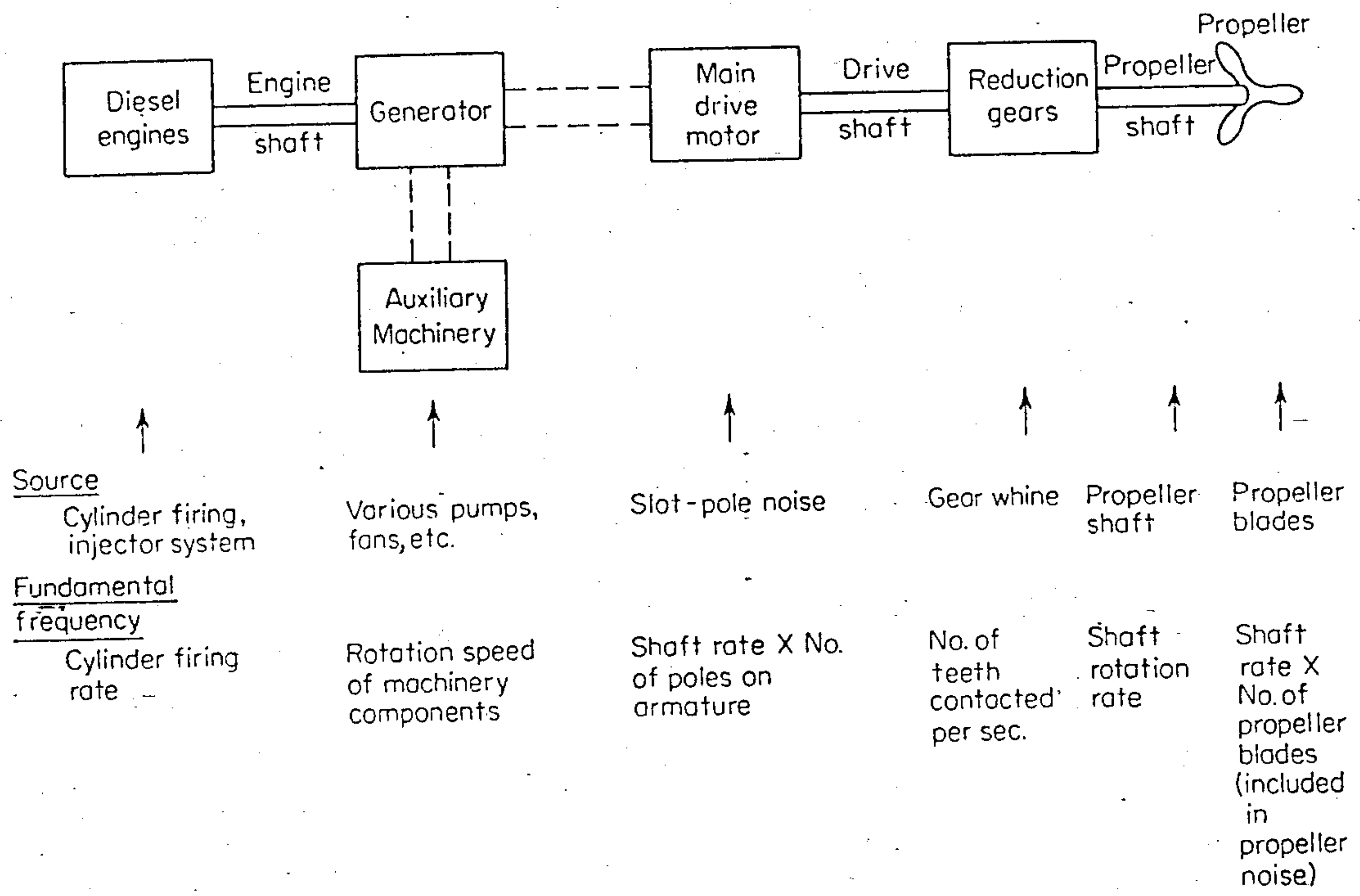


Fig. 4.2 Machinery components & noise sources on a diesel-electric vessel

flow of fluid past the moving vessel. The pressure fluctuations associated with the irregular flow may directly radiate out as sound or may excite portions of the hull into vibration.

Over much of the frequency range, the radiated noise consists of a combination of broadband noise having a continuous spectrum and tonal noise having a discontinuous spectrum containing line components (tonals) occuring at discrete frequencies. The composite spectrum is shown in Fig 4.3 . Of the three major classes of noise, machinery and propeller noise dominate the spectra of radiated noise under most conditions. The machinery noise possesses a low-level continuous spectrum with strong line components originating at the fundamental frequency and harmonics of the vibration-producing processes. The propeller noise, arising mainly from cavitation, has a continuous spectrum with a peak occuring within the frequency decade 100 Hz to 1000 Hz and 6 dB per octave roll-off on either side of the peak. The spectral characteristics of the radiated noise is a very vital parameter for target detection and feature extraction.

4.3.3 Reverberation

The ocean medium contains an innumerable variety of heterogeneities of widely varying sizes and features. They form discontinuities in the physical properties of the medium thereby acting as scatterers of the incident acoustic energy. The contributions from all the scatterers, taken in totality, is called reverberation. In active sonar , reverberation has a

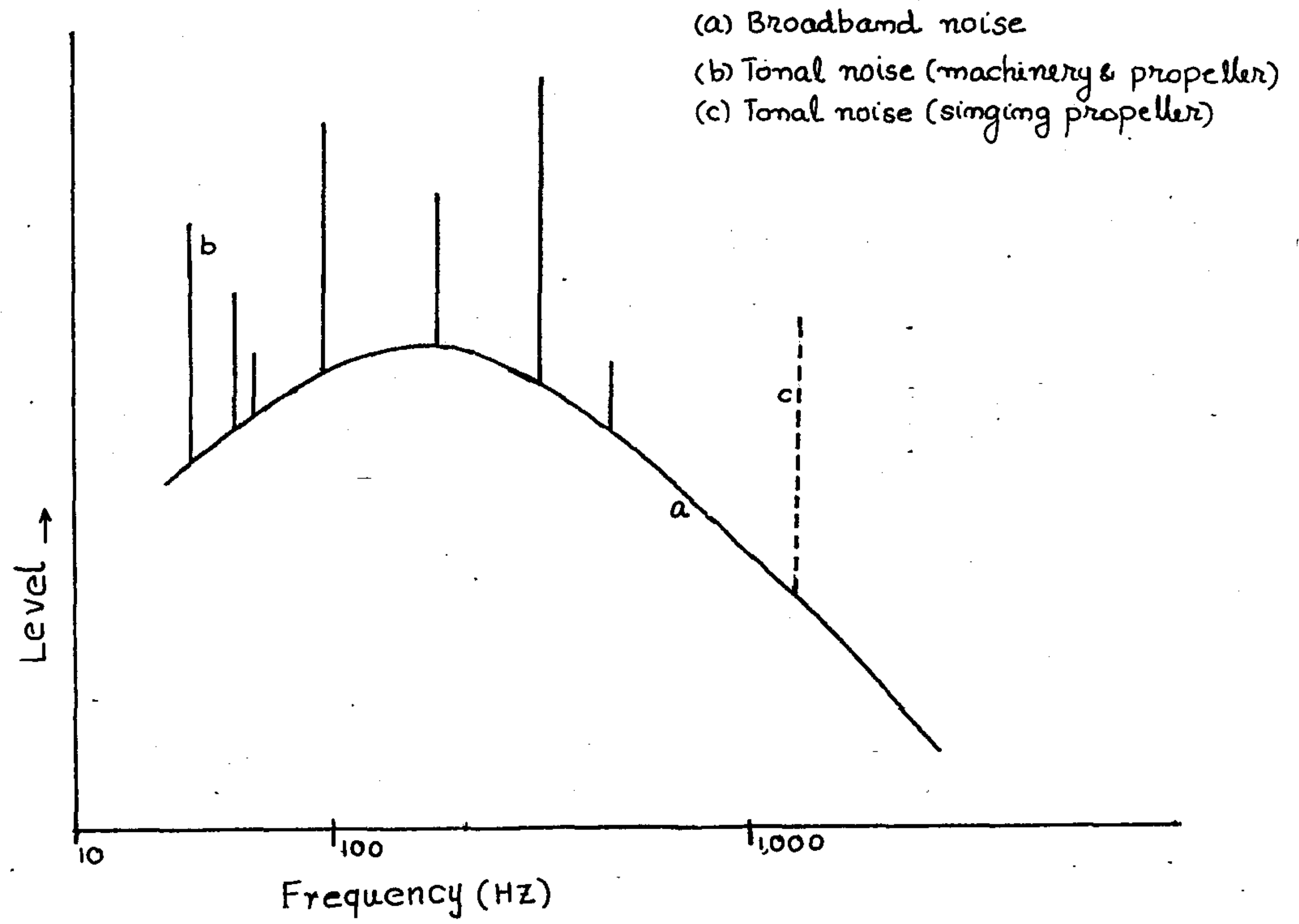


Fig. 4.3 Composite spectrum of radiated noise of submarine at high speeds

masking effect on the target echo and it imposes a primary limitation on the system performance. Marine life, suspended particles, sea-structure heterogeneities etc. cause volume reverberation while surface and bottom reverberation are produced by scatterers distributed over the sea surface and bottom respectively.

4.3.4 Self - Noise

Self-noise differs from radiated noise in that it is the noise picked up by the receiver hydrophone of the own sonar in the noise-making vessel. Since the path through which the noise reaches the hydrophone are many and varied, the relative importance of the various noise sources is different in this case. Self-noise depends greatly upon the directivity of the hydrophone, its mounting and location on the vessel. The three major sources of radiated noise discussed earlier, contribute liberally to self-noise as well. The other causes are flow-noise from the hydrophone and its support, slapping of waves against the ship's hull, impact of air bubbles, splashing, bow waves, circuit noise (hum, microphonics, transients) etc.

4.3.5 Ambient Noise

Ambient noise is the noise of the sea itself which provides a permanent background signal in sonar reception. It influences the range capability of the sonar and often is a limiting parameter. The ambient noise level, as a sonar parameter, is the intensity of the ambient background measured with a non-directional hydrophone. The ambient noise has a continuous spectrum which slopes down from the low frequency side to the high frequency side with a typical roll-off of 6 dB per octave of frequency. Hence, the noise level influences the choice of the operating frequency for optimum sonar performance. Ambient noise can be either man-made or natural. The former includes ship traffic noise, explosions, underwater communication signals etc. while the latter is caused by wind, marine life, sea-state, rain, impact of masses of water, escape of entrapped air bubbles, thermal effects, tides and hydrostatic effects of waves, seismic disturbances, oceanic turbulances, surface waves etc.

4.4 Detection of Signals in Noise : Detection Threshold

The prime task of a sonar system, whether active or passive, is to detect within a specified duration of time, the presence of the target signal in a noisy background. This is a decision-making process and some criteria have to be fixed to base this judgement with some preassigned level of accuracy of the decision as to "target present" or "target absent". The criterion selected is the signal to noise ratio (SNR) at the input of the receiver-display-observer combination. its value at the preassigned level is called the "detection threshold" (DT).

If S is the signal power in the receiver bandwidth and N the noise power in a 1 Hz band, both referred to the receiver input, then DT = 10 log (S/N) when the decision is made under the probability criteria of detection probability p(D) and false-alarm probability p(FA) which are mutually independent. At a high threshold setting, both p(D) and p(FA) are low and only strong targets will be detected. At a low threshold, both probabilities become high and too many false alarms will be sounded.

For a fixed SNR, a given threshold setting corresponds to a particular pair of values for p(D) and p(FA) and it corresponds to a point in a graph with p(FA) and p(D) as the two axes. For a continuum of threshold settings, this point traces a curve called the "receiver operating characteristic (ROC) curve". A family of such curves for different values of detection index d (which is equal to the signal-plus-noise to noise ratio of the envelope of the receiver output at the terminals where the threshold setting is established) is given in Fig 4.4 . ROC curves are strictly determined by the probability density functions of signal and noise at the receiver output where the threshold setting is made. The conventional ROC curves apply for only certain idealized and limiting conditions of signal and noise. These conditions include a steady signal in stationary Gaussian noise, large time - bandwidth products and the requirement that only a single signal at a time be detected. When these conditions do not hold, modifications to the ROCs are necessary.

For estimating the optimum detection threshold for a particular sonar, acceptable values for p(D) and p(FA)

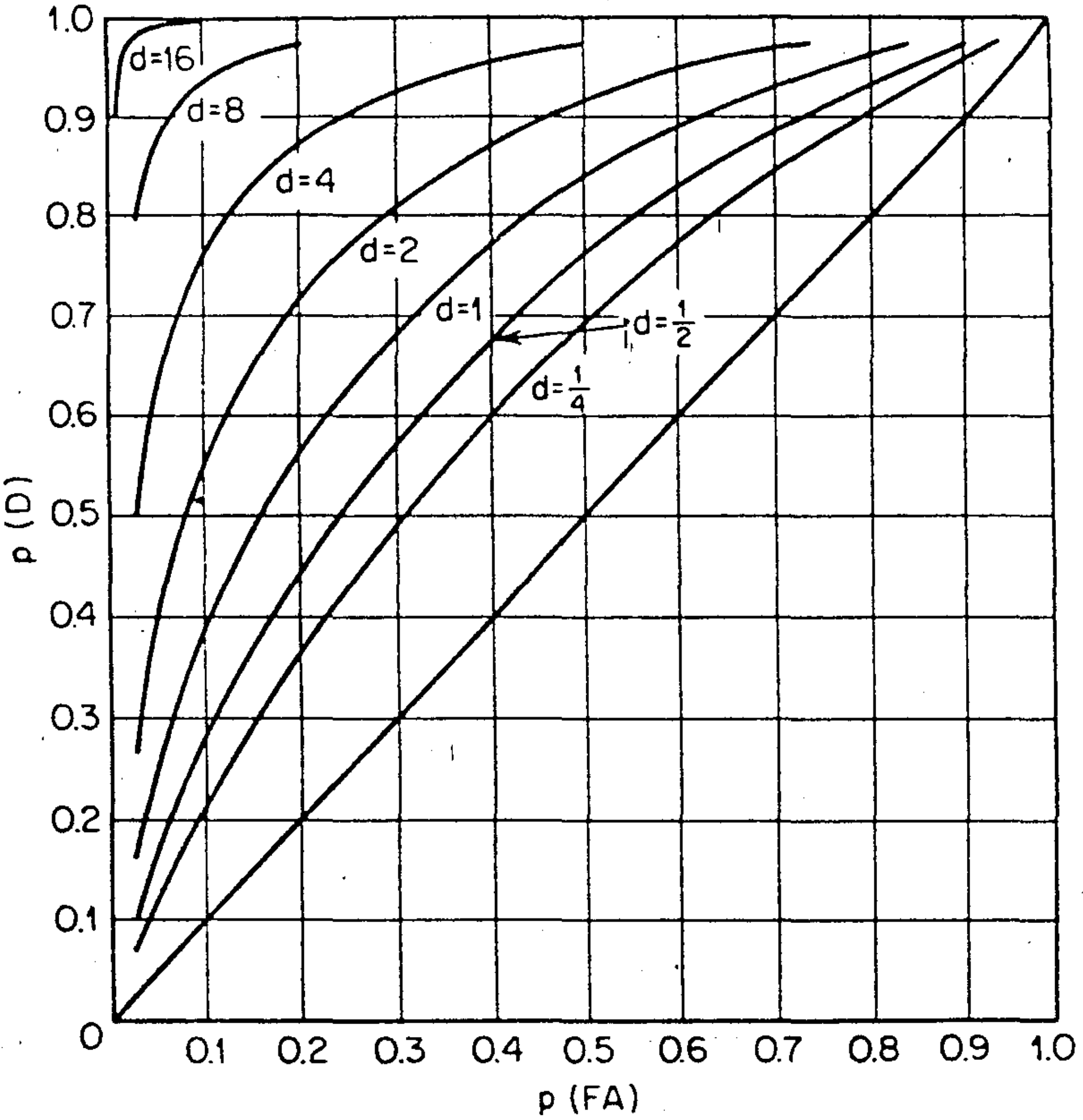


Fig. 4.4 ROC curves of p(D) against p(FA) with d as a parameter

have necessarily to be decided upon first, on a realistic basis. In a practical sonar design problem, this estimation is a rather difficult task. Most often, selection of p(D) and p(FA) are based on experience, intuition and a clear understanding of the use to be made of the system being designed. Also, when a human observer is involved in the decision process, the threshold criteria are often ill-defined and the detection threshold is consequently uncertain.

4.5 Sonar Signal Processing

This refers to the operations performed in the time - frequency domain for extracting a desired signal from a masking random noise background whose spectrum overlaps that of the signal. In such a case, the statistical properties of the signal play a key role in deciding the features of the processor. Also, it is necessary to determine how much noise may be accepted and how much signal energy may be rejected to achieve the desired result.

A sufficient statistics for detection is the "likelihood ratio" defined as the ratio of the conditional probability density of the received data vector when the signal is present to that when the signal is absent. In other words, it is the ratio of the probability that a given input amplitude represents signal plus noise (signal present) to the probability that the input represents noise alone (signal absent). The optimum processor structure is one which maximizes the likelihood ratio.

In active sonar, the signal processors are broadly classified as incoherent and coherent. For a given time - bandwidth product, the processing gain of an incoherent processor is generally inferior to that of a coherent processor. For a stationary white Gaussian noise background, the optimum processor is a "matched filter" which is equivalent to a cross - correlator [30].

In passive sonar, a noise - like signal is to be detected in a noise masking background and the signal contains both coherent and incoherent components. If the signal and noise are Gaussian random processes with known spectra, the optimum processor is some form of energy detector.

4.6 Neural Networks for Target Recognition

Target recognition is a problem which involves extraction of critical information from complex and uncertain data. Recognition of targets with fixed signatures in stationary backgrounds is a straightforward task for which numerous effective techniques have been developed [40]. If the target signatures and the backgrounds are variable in a limited or known manner, more complex techniques such as rule - based methods can be employed. But when the target signatures or the backgrounds vary in an unlimited or unknown manner, the above approaches are insufficient [31].

Target recognition needs methods to represent targets and backgrounds that are adequately descriptive and robust to variations in signature and environment. Neural networks offer potentially powerful collective - computation techniques for designing special - purpose hardware which, through powerful learning algorithms, are capable of implementing robust methods for target recognition. Neural network technology provides expert - system capabilities for automatic integration of a priori knowledge about target signatures and backgrounds so as to enhance the effective target recognition performance.

4.7 Conclusion

The survey of the Sonar Detection problem clearly brings out the complexity and diversity of the parameters involved. As such, it is felt that a multilayer neural network with backpropagation learning is necessary to handle the complex problem of Sonar Detection. The chapter to follow, therefore, demonstrates the implementation of a multilayer neural network that uses the spectral components of sonar noise in various frequency bands as its input parameters.

CHAPTER 5

NEURAL NETWORK DESIGN AND IMPLEMENTATION

The application of a massively parallel learning network to the passive detection of a sonar target (viz.a surface ship) is discussed in this chapter. The task comprises the following activities.

(a) Recording of the sea noise with and without the target ship in the vicinity of the recording platform.

(b) Preprocessing of the recorded data to generate the network training patterns.

(c) Design of the neural network.

(d) Selection of a learning algorithm.

(e) Generation of the simulation software for the neural network.

(f) Teaching the network using the training patterns and the selected algorithm.

(g) Generation of test patterns using steps (a) and (b) for different targets in different geographical locations.

(h) Testing the target detection capability of the network by presenting the test patterns to it.

These processes are pictured as a block diagram in Fig 5.1

5.1 Recording of Sea Noise

For recording of the sea noise, the "SUBER" Remote-Controlled Acoustic Data Acquisition Module was used. The

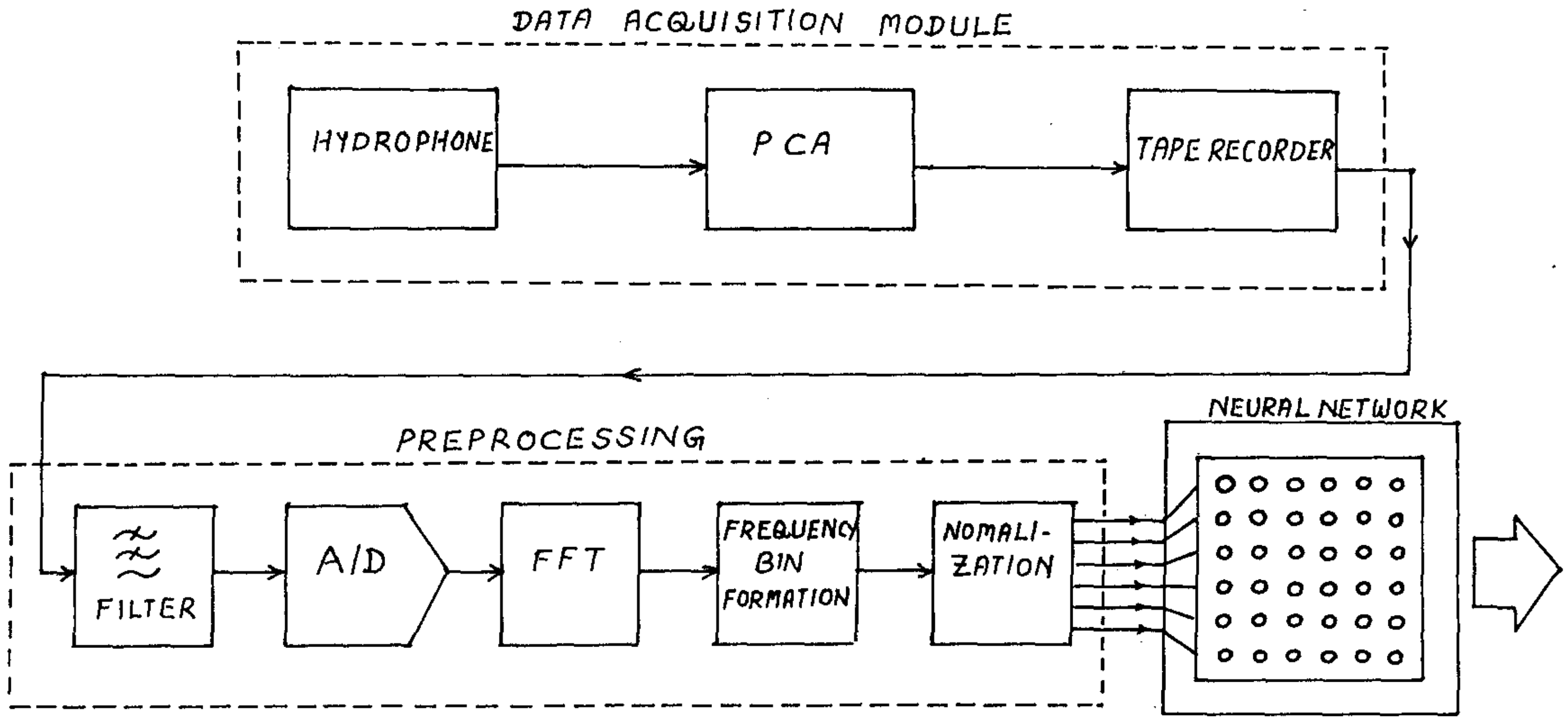


FIG. 5.1 Neural network scheme for sonar target detection

electrical signal from the hydrophones corresponding to the underwater sound they picked up is fed to the instrumentation tape recorder after amplification by the precision conditioning amplifier (PCA).

Recordings with and without target ships sailing in the vicinity were taken at different geographical locations using the data acquisition modules deployed at these locations. The set-up for recording of the sea noise is shown in Fig. 5.2 .

5.2 Preprocessing of the Data

The front-end processing is an essential element to any neural network technique. If given non-representative or inadequate training data, any neural network paradigm will perform poorly. Neural networks do provide a novel method of abstracting feature information into a distributed encoding. They do not, however, by-pass the critical stage in any pattern recognition task of adequately defining the salient and characteristic features of the data.

The preprocessing employed in the present work relies on the extraction of the spectral data using a Fast Fourier Transform (FFT) computation on the A/D converted signal obtained from the sea noise recordings. This technique shows the clustering properties of the spectral components better than more conventional coding methods and thus provides a more useful representation on which the classifier can train. It is also a

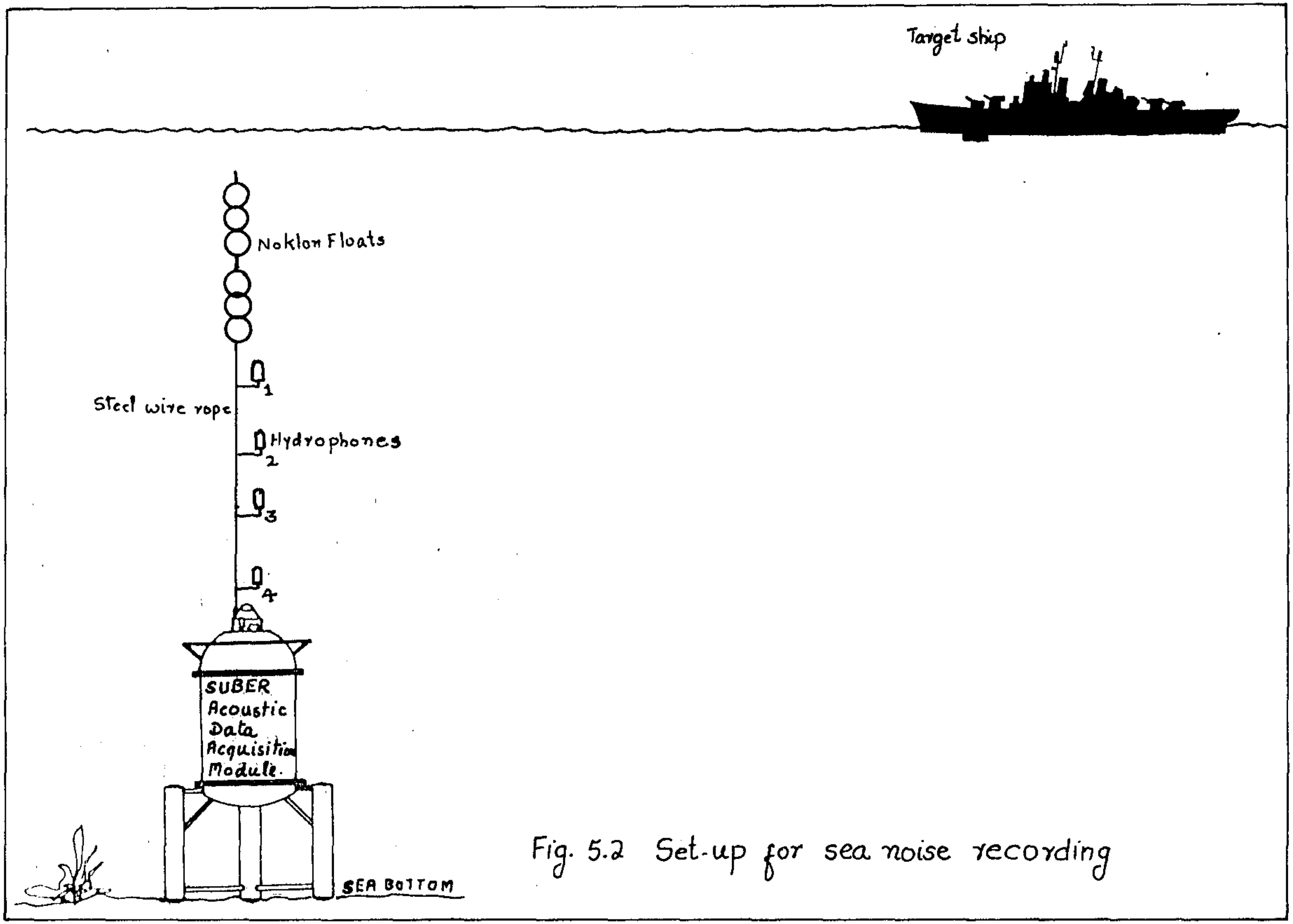


Fig. 5.2 Set-up for sea noise recording

fast, reliable and well-supported technique [6].

The analog output signal from the tape recorder corresponding to the recorded data was low-pass filtered upto a cutoff frequency of 750 Hz and subsequently amplified by 20 dB.

The analog to digital converter (A/DC) resolution was 12 bit and a sampling frequency of 2 kHz was selected since the prominent frequency components in the radiated noise of the target are expected only upto 1000 Hz.

A 1K - point FFT was computed on the digitized time-domain data samples from the A/DC to convert them into frequency domain samples which correspond to the spectral components in the signal. A twenty-point vector was formed from these spectral components in the selected bandwidth of the signal by segregating them into twenty consecutive frequency bins with twenty Fourier coefficients in each bin. Since each point in the vector was formed by adding up the squared coefficients in each bin, its value gives a measure of the energy contained in the respective bin. The above vector, after normalization, was presented as the input to the neural network.

Input vectors were generated from time-strips of sea noise recordings with and without targets, made at different geographical locations. Recordings made at one of these locations were used to generate the training vector patterns while those made at other locations were used to generate the testing ones.

5.3 Neural Network Design

A multi-layer perceptron was used for the experimentation. In the light of the discussion in section 3.4.3, a three-layer network configuration with the number of neurons in the middle layer being more (viz. exceeding by two) than three times that in the input layer was selected. The type of preprocessing proposed in section 5.2 needs 20 neurons in the input layer. Since separate neurons were assigned for "target present" and "target absent" outputs, a network with 20, 62 and 2 neurons respectively for the input, middle and the output layer resulted. This is designated as a 20 - 62 - 2 network.

The performance of two more network topologies viz. 20 - 62 - 1 and 20 - 11 - 2 also was studied for comparison purposes.

The above three networks are shown in Fig 5.3, 5.4 and 5.5 respectively.

5.4 The Learning Algorithm

The backpropagation algorithm was used since it is most suited for multi-layered networks. Using the momentum method and the exponential smoothing method, networks were separately trained and their performance was compared.

For every pattern vector presented, each neuron

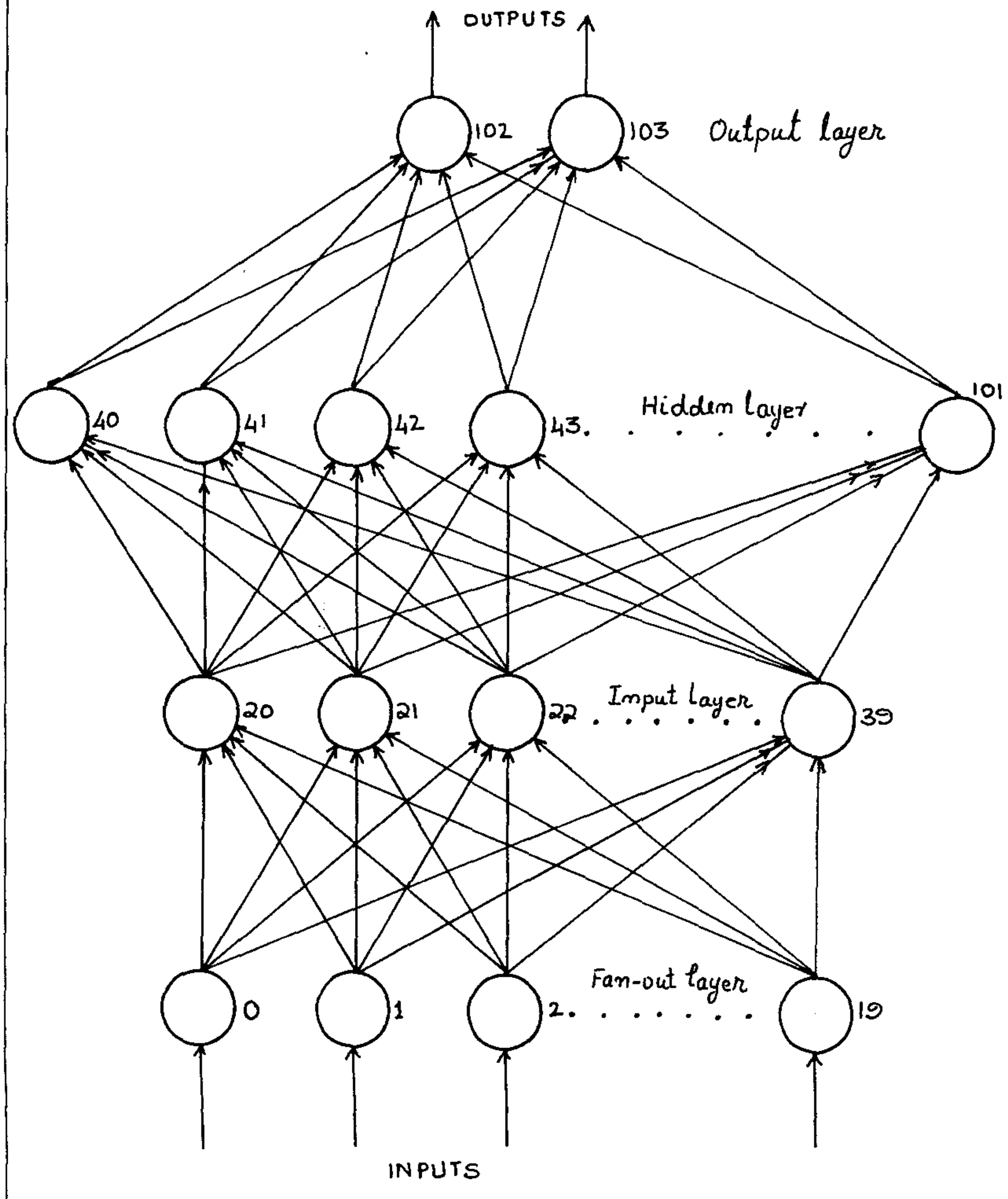


Fig. 5.3 The 20-62-2 Neural network

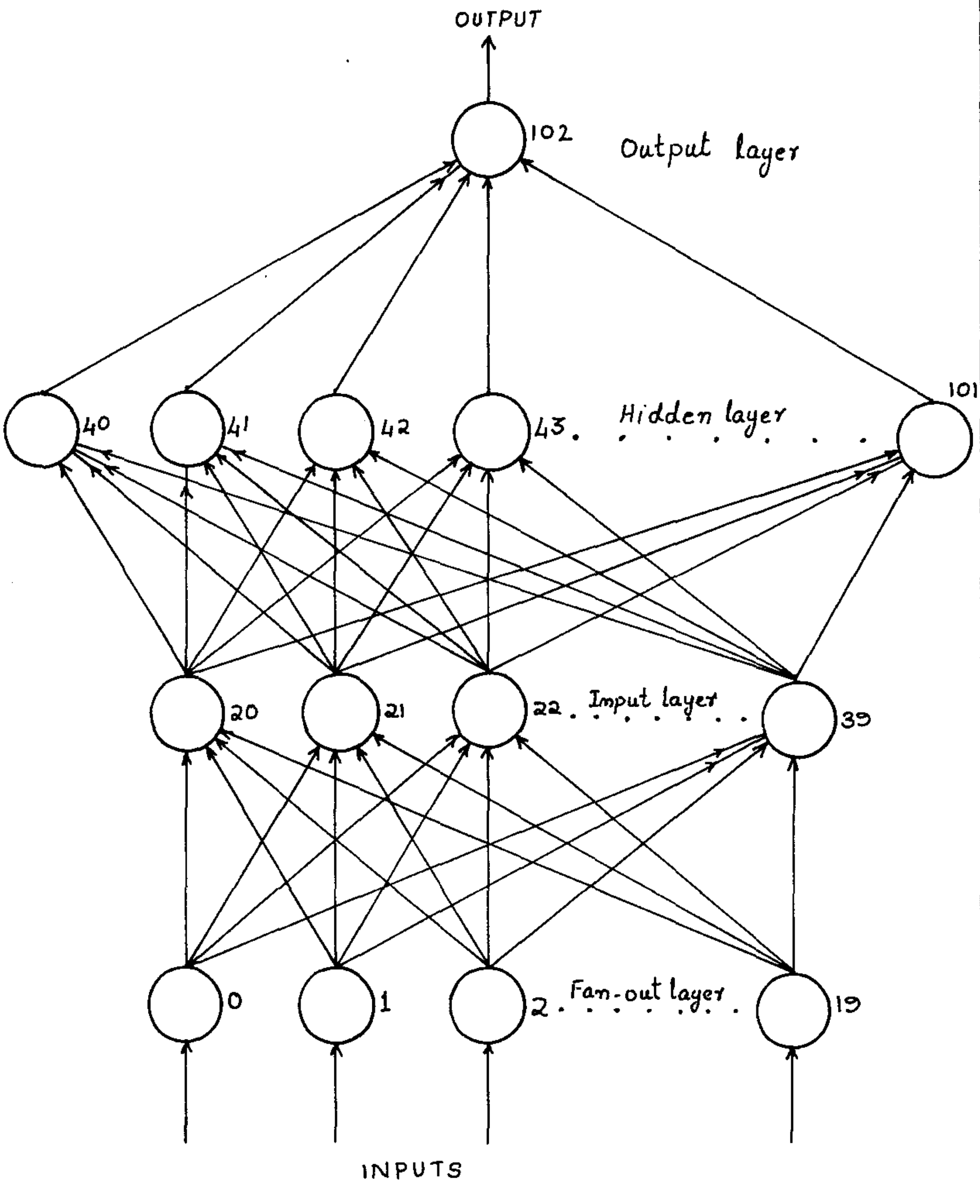


Fig. 5.4 The 20-62-1 Neural network

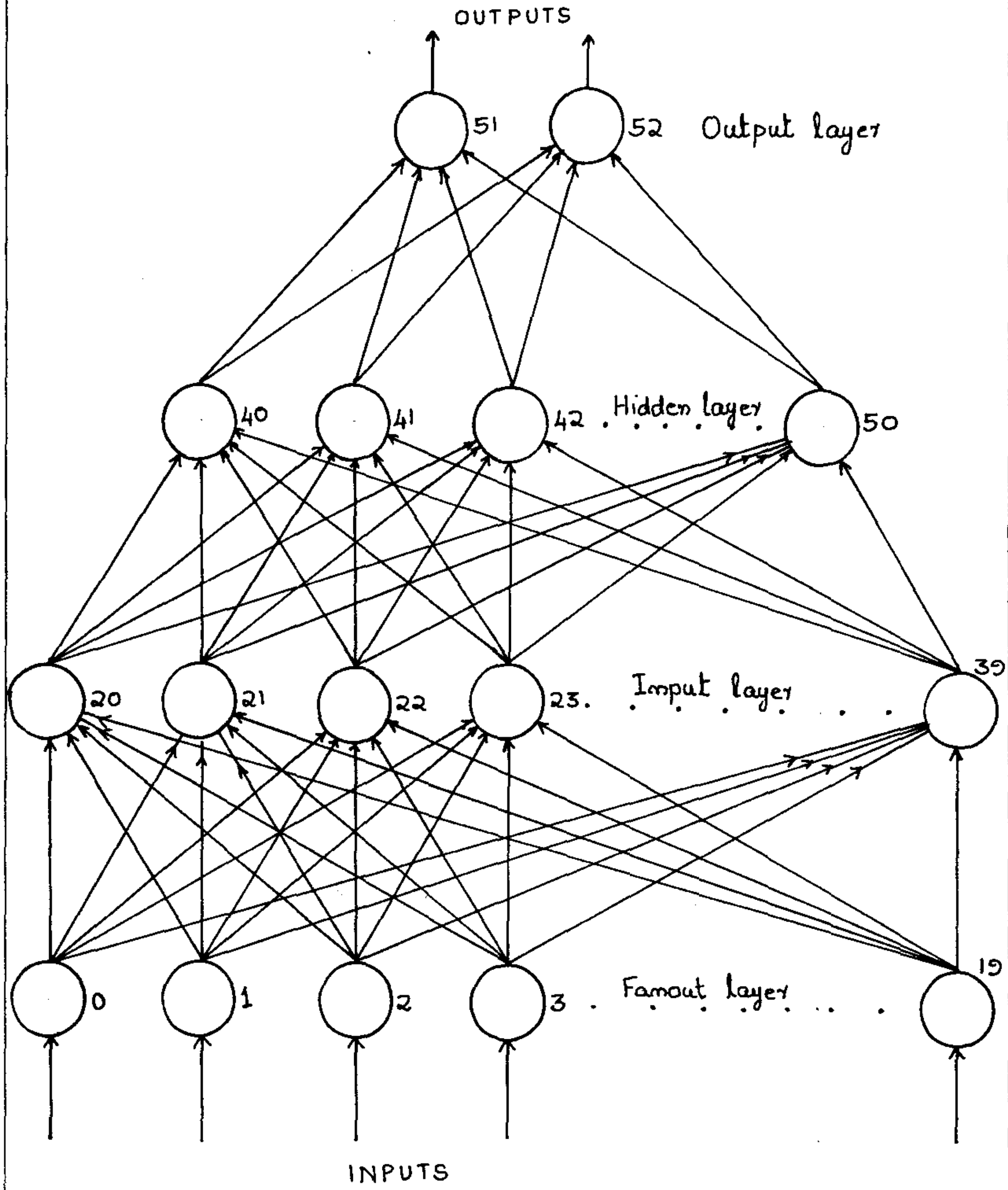


Fig. 5.5 The 20-11-2 Neural network

computes the net value obtained from equn. (3.1) modified by a bias as mentioned in section 3.4.1 . The output from the neuron was obtained by thresholding the above with the sigmoidal nonlinearity (ref. equn. (3.3) with a=1).

Depending upon the learning algorithm chosen, either equn. (3.20) or (3.21) was used to modify the weights associated with the neurons in various layers. The weight error derivative δ_j in these expressions was calculated using equn. (3.15) and (3.17) respectively for neurons in the output layer and those in all the other layers.

5.5 Simulation Software Structure

Though neural computing is basically a parallel distributed processing procedure, the mathematical operations involved in it can easily be carried out on conventional computers. This facility for computer simulation of neural networks offers phenomenal possibilities for experimentation.

The software that was designed for the simulation work under discussion suits multi-layered feed-forward networks with any topology (limited by the memory available in the computer), any learning algorithm and any number of training/testing patterns.

The network topology is characterised by the following quantities which are to be specified in the network specification file.

(a) Total number of neurons in the network (including the fan-out neurons at the network input).

(b) Number of input neurons.

(c) Number of output neurons.

(d) First weight to last weight associated with each of the neurons.

In the network learning phase, all the input training patterns were specified into the software as files with details as to the number of files, file size and the total number of patterns. Output patterns (ie. target vectors) were specified as an array. While passing through this phase, the network gradually converges and its weights and biases attain their generalised values at the end of this phase.

In the network testing phase, patterns with features (to the extent of the information as to whether target is present or absent) unknown to the network were applied to it one after the other and the corresponding output response was compared with the correct feature.

5.5.1 The Learning Phase

This comprised the following steps:

(a) Specify the network topology, training pattern details and the values of the appropriate coefficients used in the learning algorithm.

(b) Initialize all weights, biases and their derivatives to zero.

(c) Specify the number of iterations to be carried out and a starting value & the desired minimum value for the sum of square error.

(d) Either read the weights and biases already available in a file ("IPFILE") or generate them with random values. Also, store them in a file ("STARTWTS") in the latter case.

(e) Read all the training patterns viz. the input and the corresponding target patterns from the relevant files and the target array respectively.

(f) Set the control appropriately so that the training patterns are presented to the network one after the other in the same order in which they were read or in a random order.

(g) Carry out one epoch of training. This includes the following steps:

   (i) Apply one training pattern to the network.

   (ii) Compute the output of the network corresponding to the applied input pattern and the weights & biases already set in.

   (iii) Compute the error between the applied target value in step(i) and the value obtained in step (ii).

   (iv) Using the above error, calculate the error derivatives for all the weights and biases of the network.

   (v) Using the above derivatives in the learning algorithm, modify all weights and biases. Replace the earlier weights and biases by their respective modified values.

   (vi) Repeat steps (i) through (v) for the remaining patterns to complete one epoch of training and this is taken as one iteration. The weights and biases obtained at the end of every epoch serve as the starting weight and bias values for the subsequent epoch.

(h) Square the error values obtained in step (g)(iii) for one iteration and add them up to get the sum of square error. Store this value and the corresponding order of iteration in a file ("TSSFILE").

(i) If the sum of square error newly obtained in step (h) is less than its earlier value, update the latter to the new value and store the corresponding weights and biases obtained as per step (g)(vi) at the end of the relevant iteration in a file ("FINALWTS").

(j) Repeat steps (g) through (i) until the sum of square error becomes equal to or less than the minimum value specified in step (c) or the execution of the number of iterations specified therein, whichever happens earlier.

(k) Store the weights and biases obtained at the end of the last iteration in a file ("LASTWTS").

The software archicture for network learning is depicted in Fig. 5.6 through Fig. 5.10 .

5.5.2 The Testing Phase

During the learning phase, the network traverses through a gradient descent path in the error surface and at the end of this phase involving sufficient number of iterations, it converges and settles at the global minimum. With its generalized weights and biases, which correspond to the features

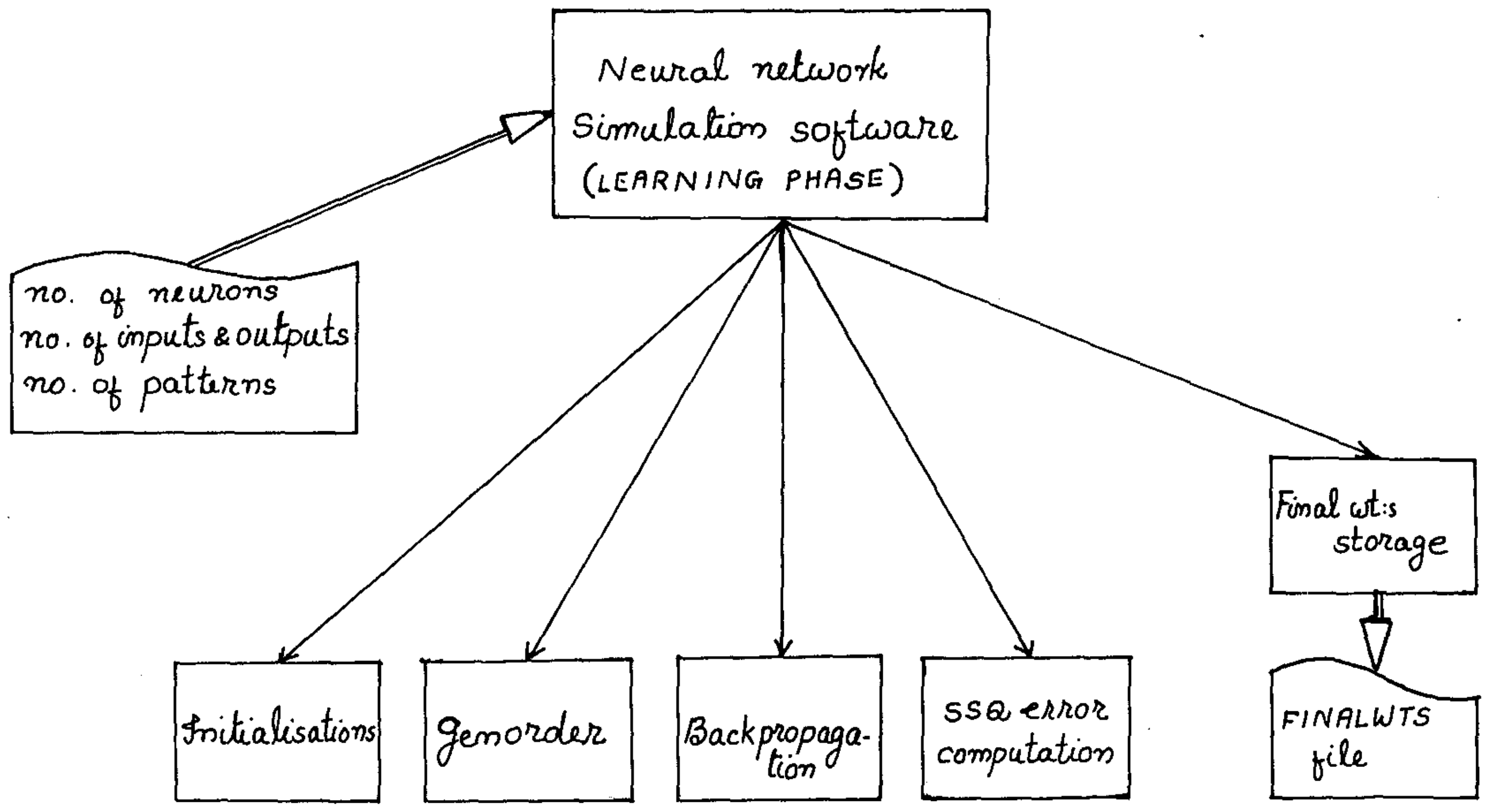


Fig. 5.6 Software architecture diagram 1

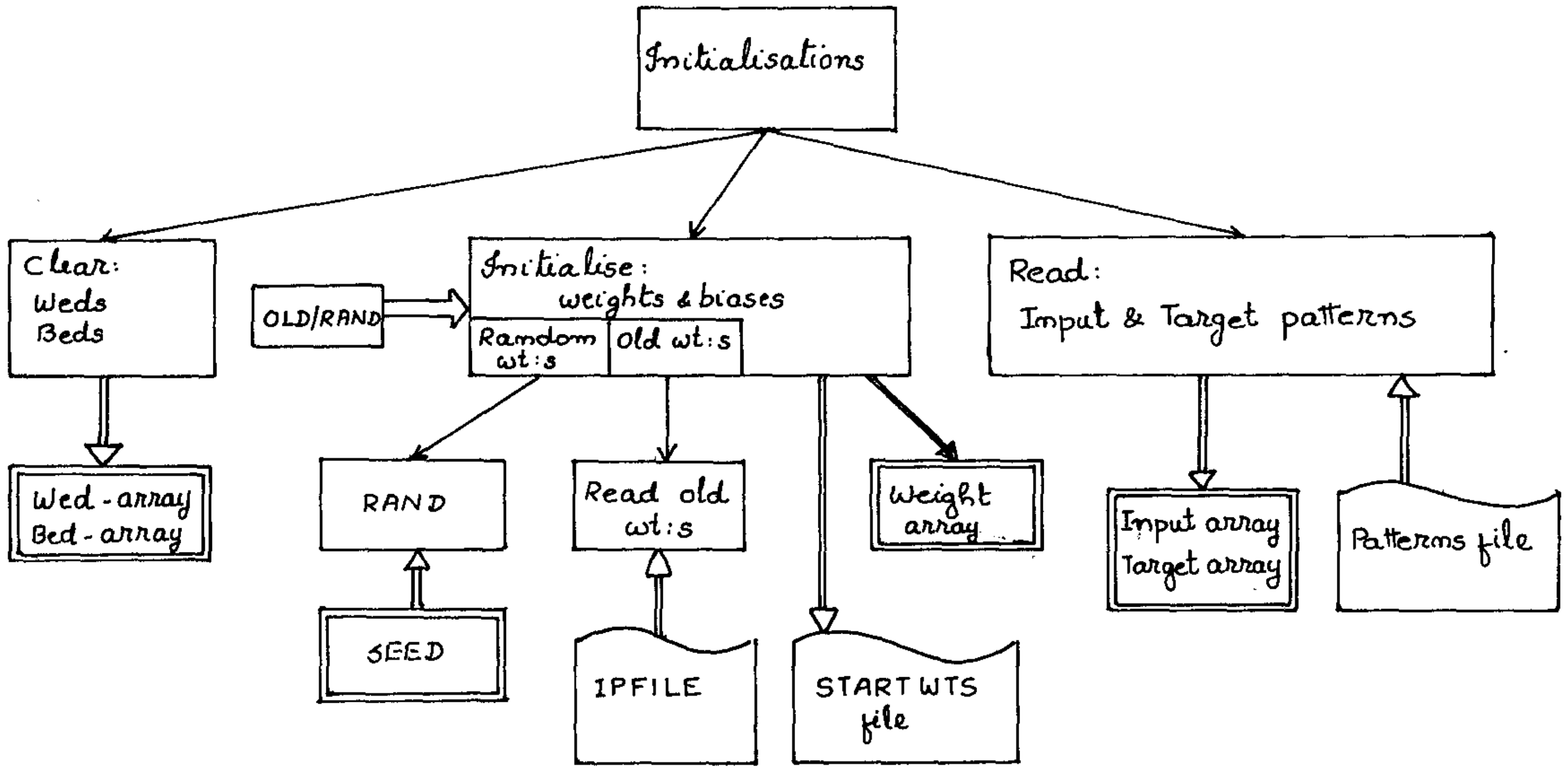


Fig. 5.7 Software architecture diagram 2

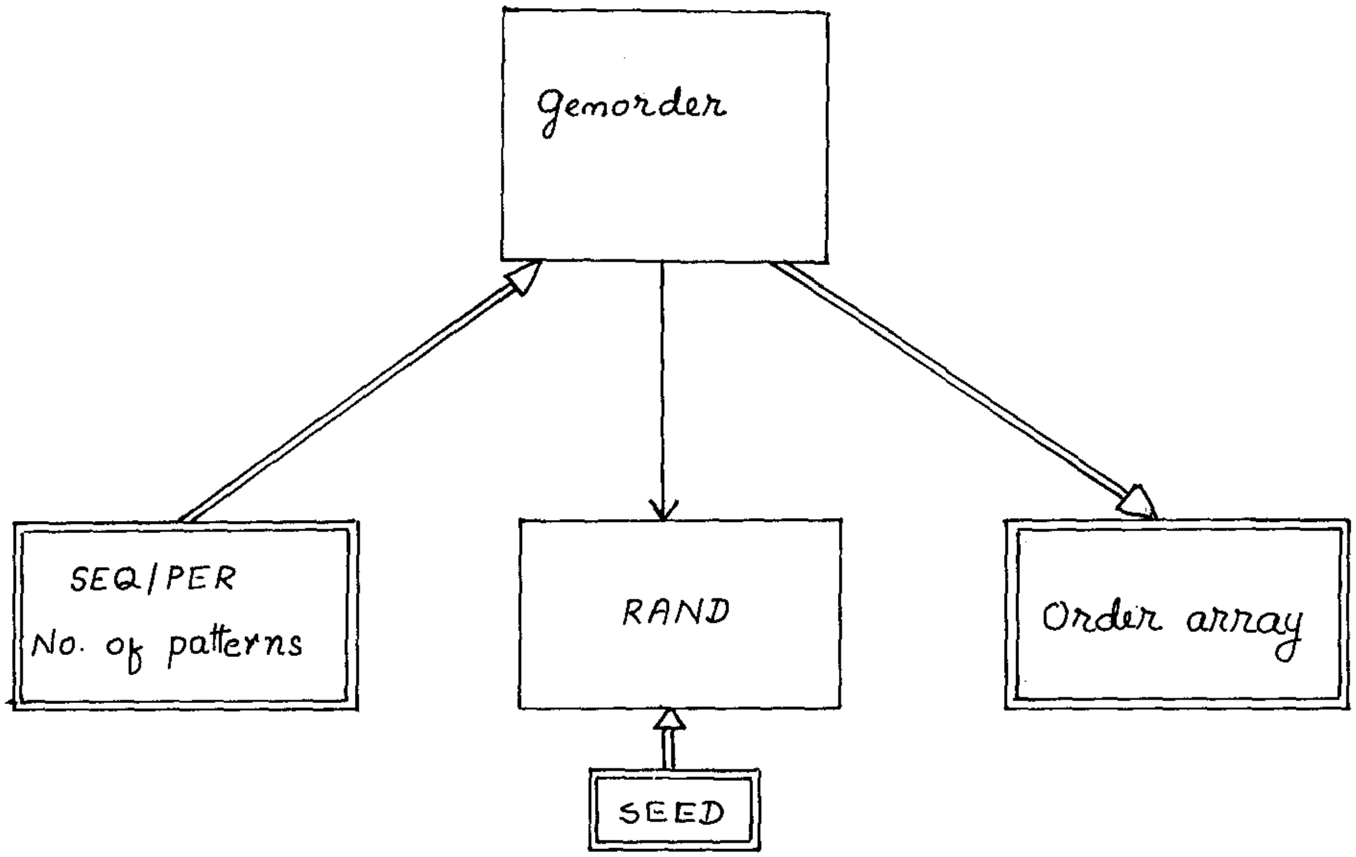


Fig. 5.8 Software architecture diagram 3

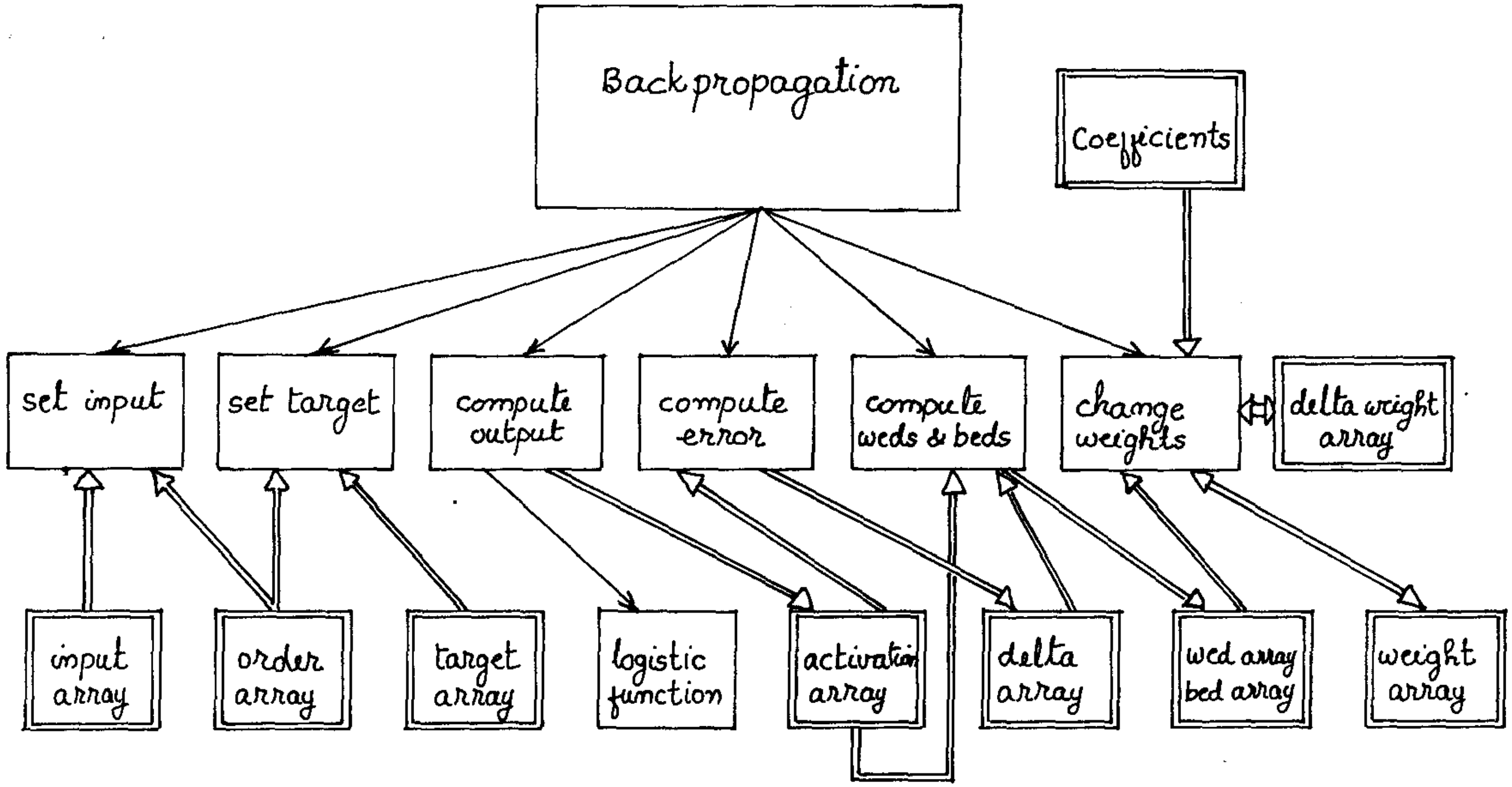


Fig. 5.9 Software architecture diagram 4

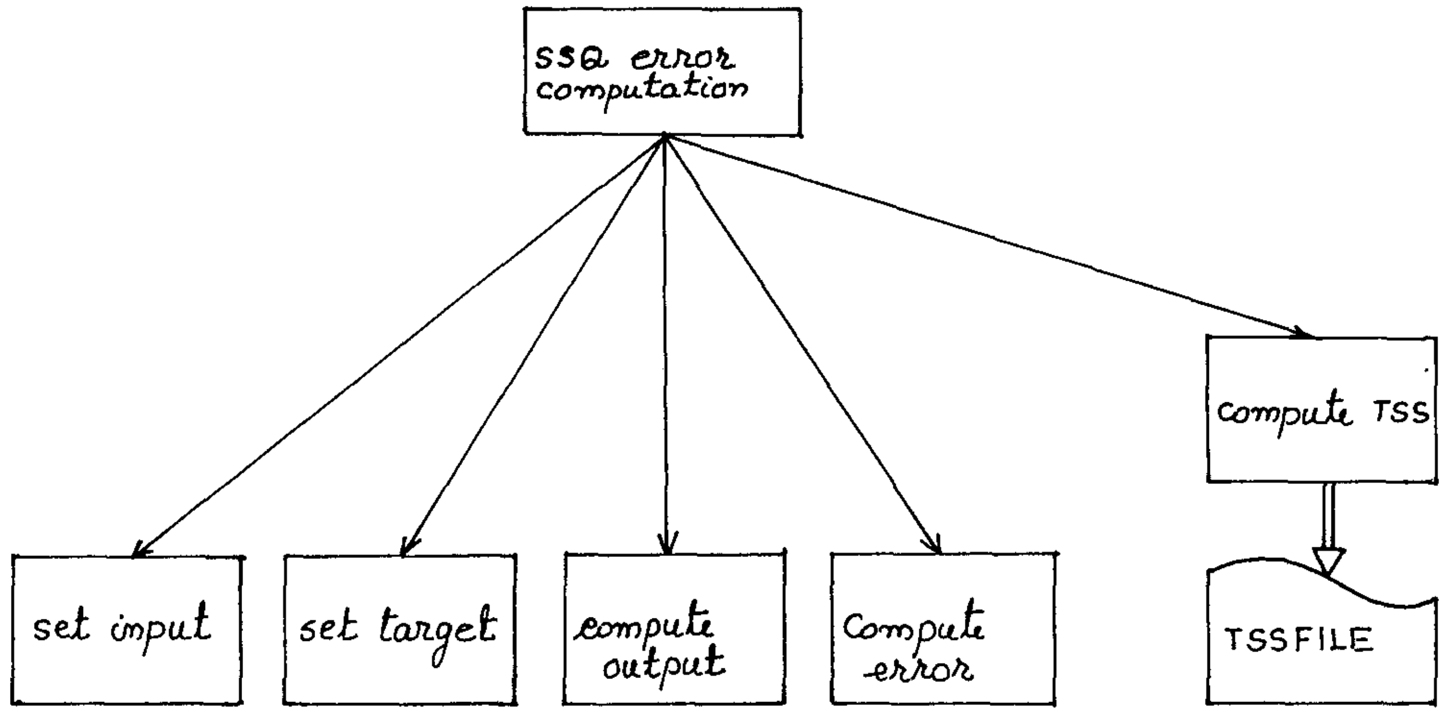


Fig. 5.10 Software architecture diagram 5

latent in the training patterns, the network can now detect the presence of these features in the unknown patterns presented to it. This process is carried out in the testing phase which is a non-iterative, one-shot, forward-pass computation.

This phase involved the following steps:

(a) Specify the network configuration in the manner mentioned at the beginning of section 5.5 .

(b) Specify all the input patterns for testing (comprising only input vectors and no target vectors) as done earlier for the training patterns.

(c) Read all the test patterns.

(d) Load in to the network, the generalized weights and biases from a file ("USEDWTS") which points to the contents of the file "FINALWTS" which was created during the learning phase.

(e) Set one test pattern at the network input.

(f) Compute the network output.

(g) Classify the network output with "target present" or "target absent" labels by applying appropriate thresholds to these outputs.

(h) Repeat steps (e) through (g) for all test patterns.

The software structure for network testing is given in Fig. 5.11 .

The source programme for the neural network simulation was developed in C language and it was run on CYBER 180/830 mainframe computer.

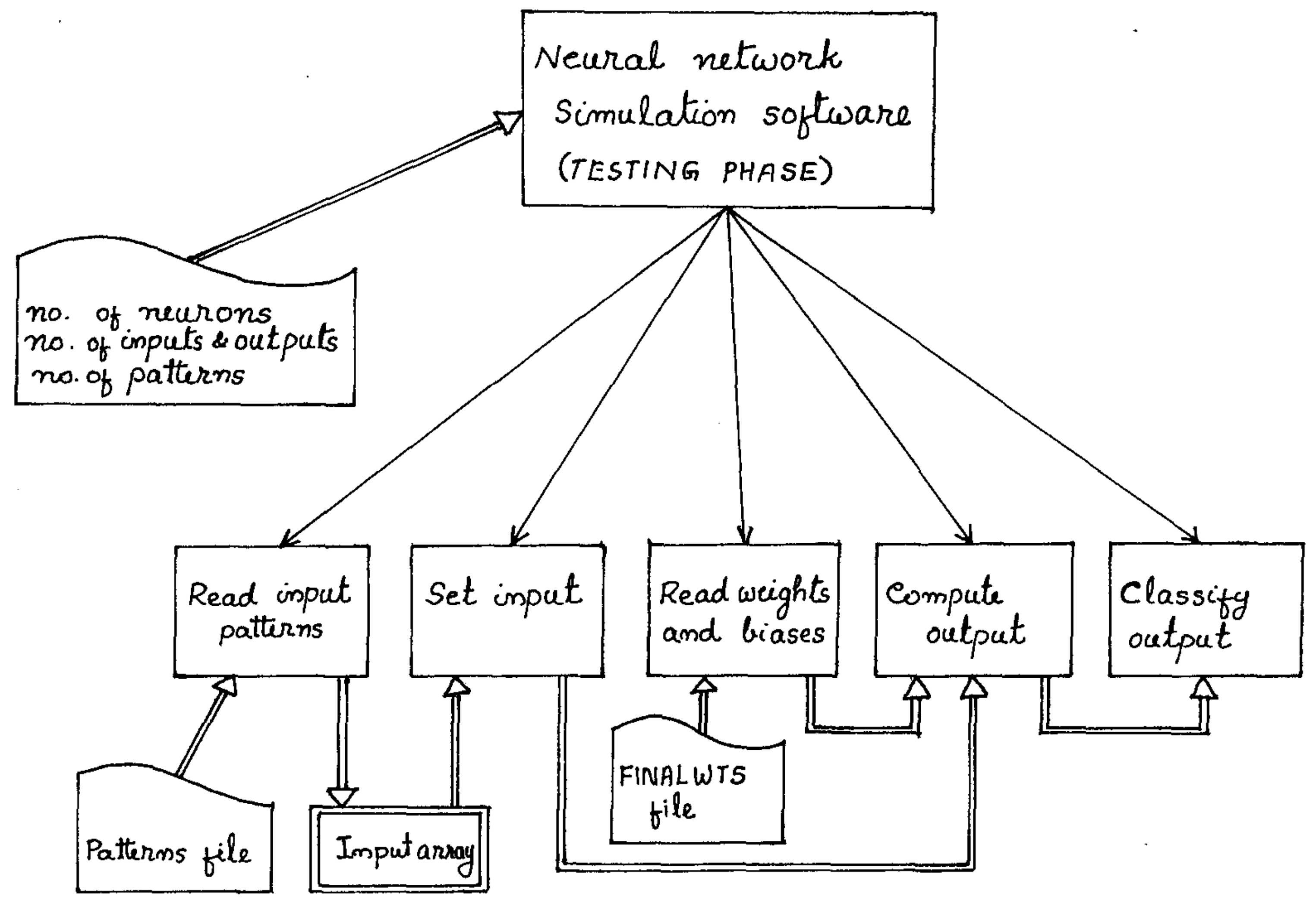


Fig. 5.11 Software architecture diagram 6

5.6 Classification Labeling

Since ambient noise as well as target-radiated noise was used for both training and testing the network and as the problem is of two-class nature, four combinations are possible for the above. These were segregated under three classification labels as shown in Table 5.1 below. Out of the

TABLE 5.1 Classification Label

Test Input	Classification Result	Classification Label
Ambient noise	Target Present	False Alarm
Radiated noise	Target Absent	Miss
Ambient noise	Target Absent	Correct Classification
Radiated noise	Target Present	Correct Classification

total testing patterns, the percentage of correct classification obtained under these labels for each network-algorithm combination was used to make a comparative study of their classification efficiency.

5.7 Conclusion

The preparation of data for presentation to the neural network, the design details of the network and the simulation software structure were discussed here. The test results are summarized and discussed in the chapter to follow.

CHAPTER 6

RESULTS AND DISCUSSION

Simulation details like network topologies tried, learning algorithms used and the convergence behaviour and classification capabilities of various combinations of the learning algorithm and network topology are discussed in this chapter.

6.1 Input Patterns

Sea noise recordings with and without target, taken exclusively at one geographical location (viz. Cochin) were used to generate the training patterns. In this connection, 40 vectors corresponding to ambient noise alone and 32 vectors corresponding to target noise were used. During every iteration, these two sets of patterns were presented to the network in an interlaced manner.

For checking the target detection capability of the network, 248 test vectors were used. Out of these, 196 vectors were generated from ambient noise and 52 vectors from target noise which were recorded at other geographical locations (viz. Andaman and Goa).

6.2 Network Topology

Three network configurations were tried viz. 20-62-2, 20-62-1 and 20-11-2. In the first two networks, the number of neurons in the second layer is more than thrice that in the first layer whereas in the third one, the number of neurons in the middle layer is the mean of the number of neurons in the other two layers. They also differ in regard to the number of neurons in the output layer. The aim of simulating these network variants was to investigate how the above factors influence the network learning behaviour and their classification performance.

6.3 Learning Algorithms

Each of the network was separately trained using two variants of the BP algorithm viz. momentum method and exponential smoothing. In the former, for all the network configurations, the learning rate and momentum coefficient were given the values 0.75 and 0.9 respectively. In the latter case, the exponential smoothing coefficient was given the value 0.9, keeping the learning rate coefficient unaltered. These set of values were found to provide the fastest learning rate without causing oscillation.

6.4 Neural Network Training

Before the commencement of the network training, all the weights and biases were initialized to random values in the range from -0.1 to +0.1 . The training patterns were then presented to the network one after the other and the weights and biases were changed after every presentation according to the learning algorithm used. This process was continued till all the training patterns were exhausted (ie. completion of one epoch or iteration). After every iteration, the sum of the square of the errors obtained while presenting all the 72 patterns during that epoch was computed. If it was more than the lower limit set viz. 0.01 in the present case, the iteration was continued further until this limit is reached or the prescribed number of iterations are carried out, whichever is earlier. The algorithms were run for a minimum of 1000 iterations in each trial.

6.5 Learning Curve

Sufficient number of iterations will inevitably have to be carried out before the network reaches its converged state where the error is the minimum-most. The quality and ingenuity of a learning algorithm is judged based on its capability to achieve network convergence with minimum number of iterations. The trend of the error during progressive iterations is a reliable index of the learning behaviour of the network. As the sum of square error is an aggregate of the errors during an epoch of training, a plot between this quantity and the corresponding order of iteration is called the "learning curve" for the relevant network - algorithm combination.

The learning curves for the three networks corresponding to the two algorithms are given in Fig. 6.1 through Fig. 6.6 .

Various results obtained in the simulation studies are summarily presented in table 6.1 .

6.6 Observations

Since the simulation studies carried out in the present work cannot be claimed as rigorously exhaustive, a generalization of the results obtained therefrom doesnot bear much relevance in the strictest sense. However, the observations made within the limited scope of the experiments are as listed below:

(a) As can be seen in all the learning curves, the maximum value attained by the sum of square error is relatively moderate. This is attributed to the type of preprocessing employed.

(b) Convergence of the learning algorithm is deeply influenced by the initial weights and biases (which, in turn, are determined by the seed value given to the random number generation routine).

(c) The rate of convergence is dependent on the learning rate coefficient η , momentum coefficient α , and exponential smoothing coefficient β . For faster convergence, both η and α should be high and for maximum smoothing, β should be high. But, with high η , the algorithm either oscillates or diverges. A high β ensures smoother learning which consequently takes more time to converge and this is evident

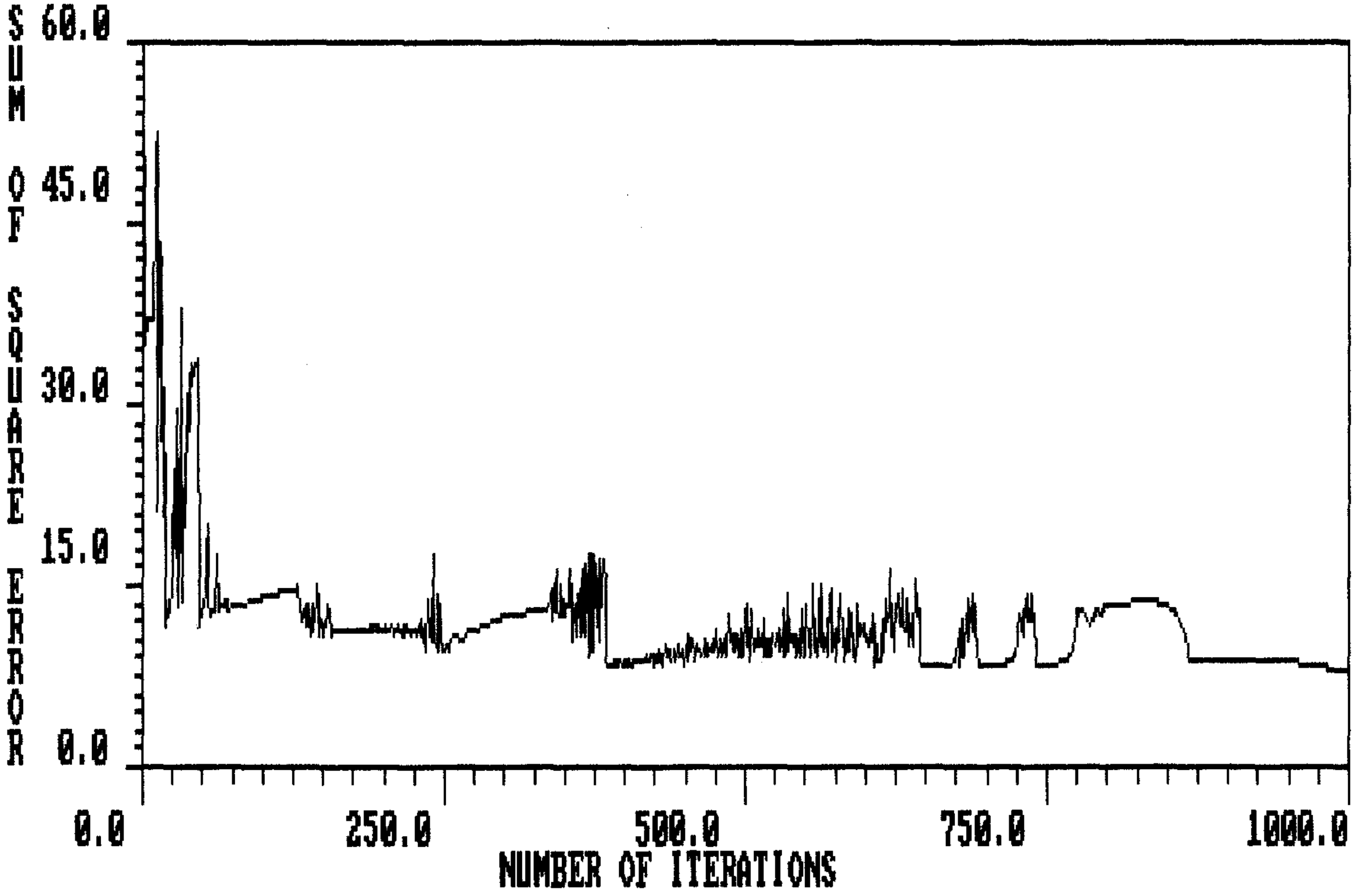


Fig. G.1 Neural Network Learning Curve
Network Topology : 20-62-2 Learning Algorithm : BP(Momentum)

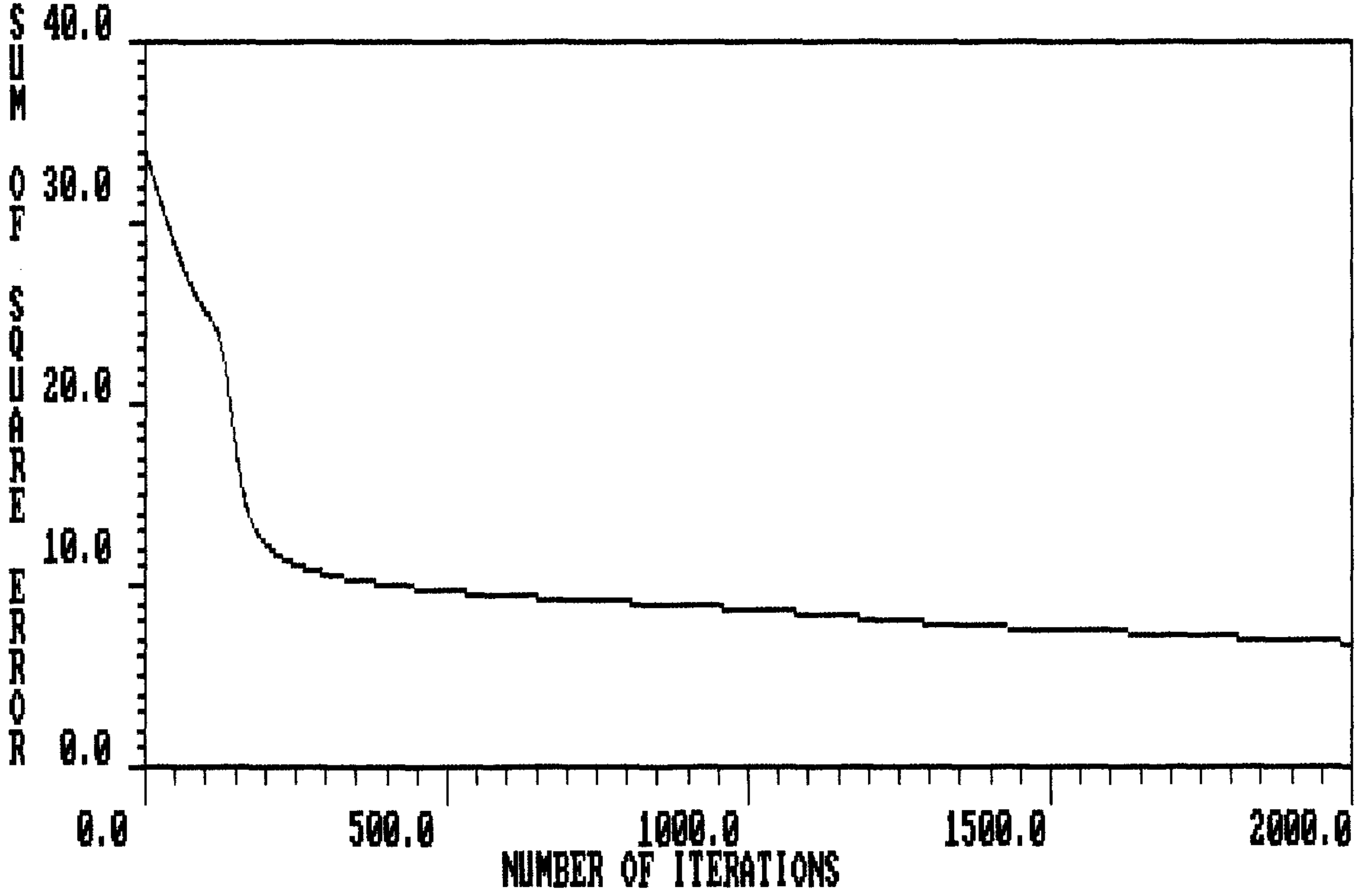


Fig. 6.2 Neural Network Learning Curve
Network Topology : 20-62-2 Learning Algorithm : BP(Exp. Smoothing)

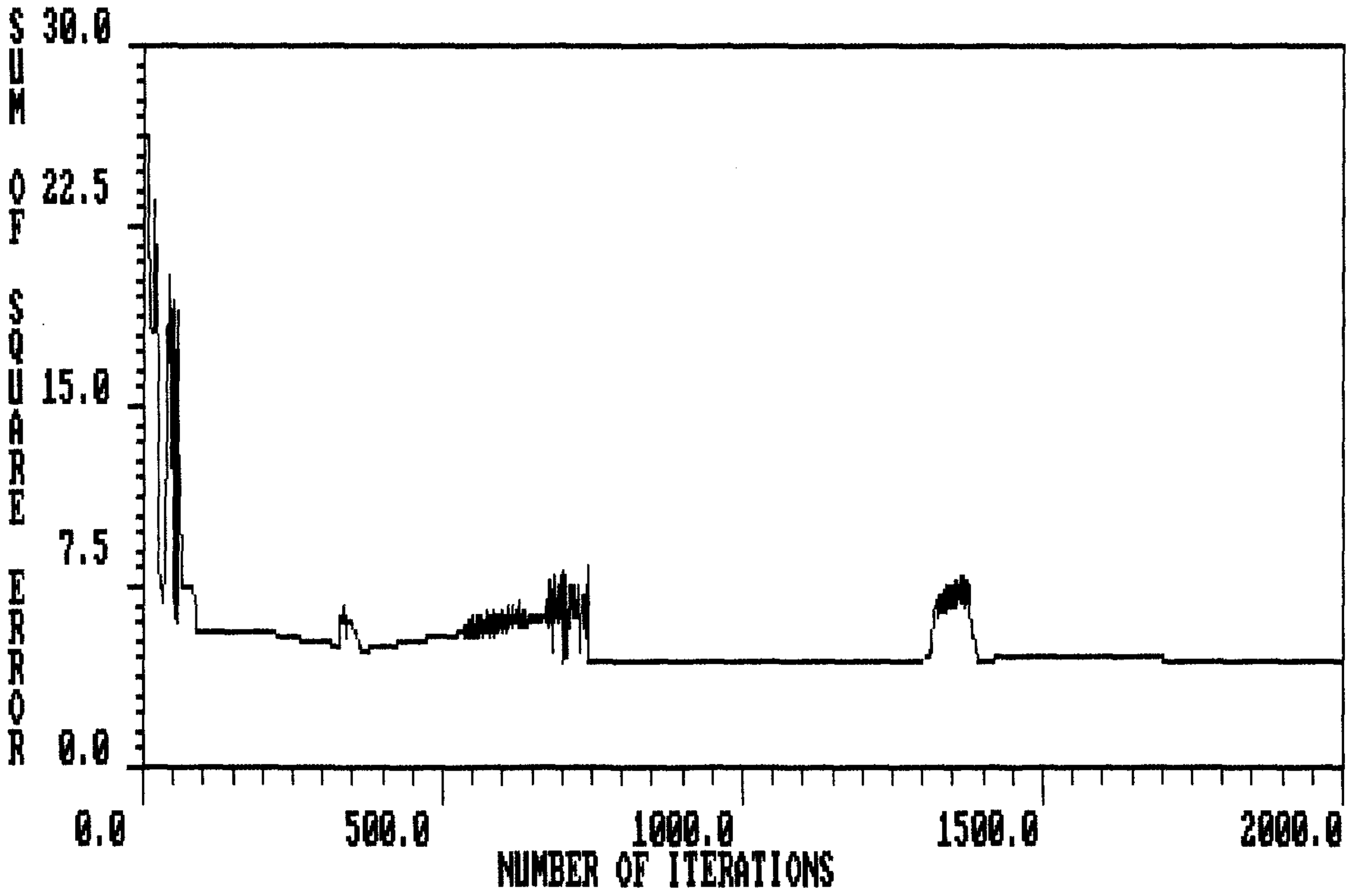


Fig. 6.3 Neural Network Learning Curve
Network Topology : 20-62-1 Learning Algorithm : BP(Momentum)

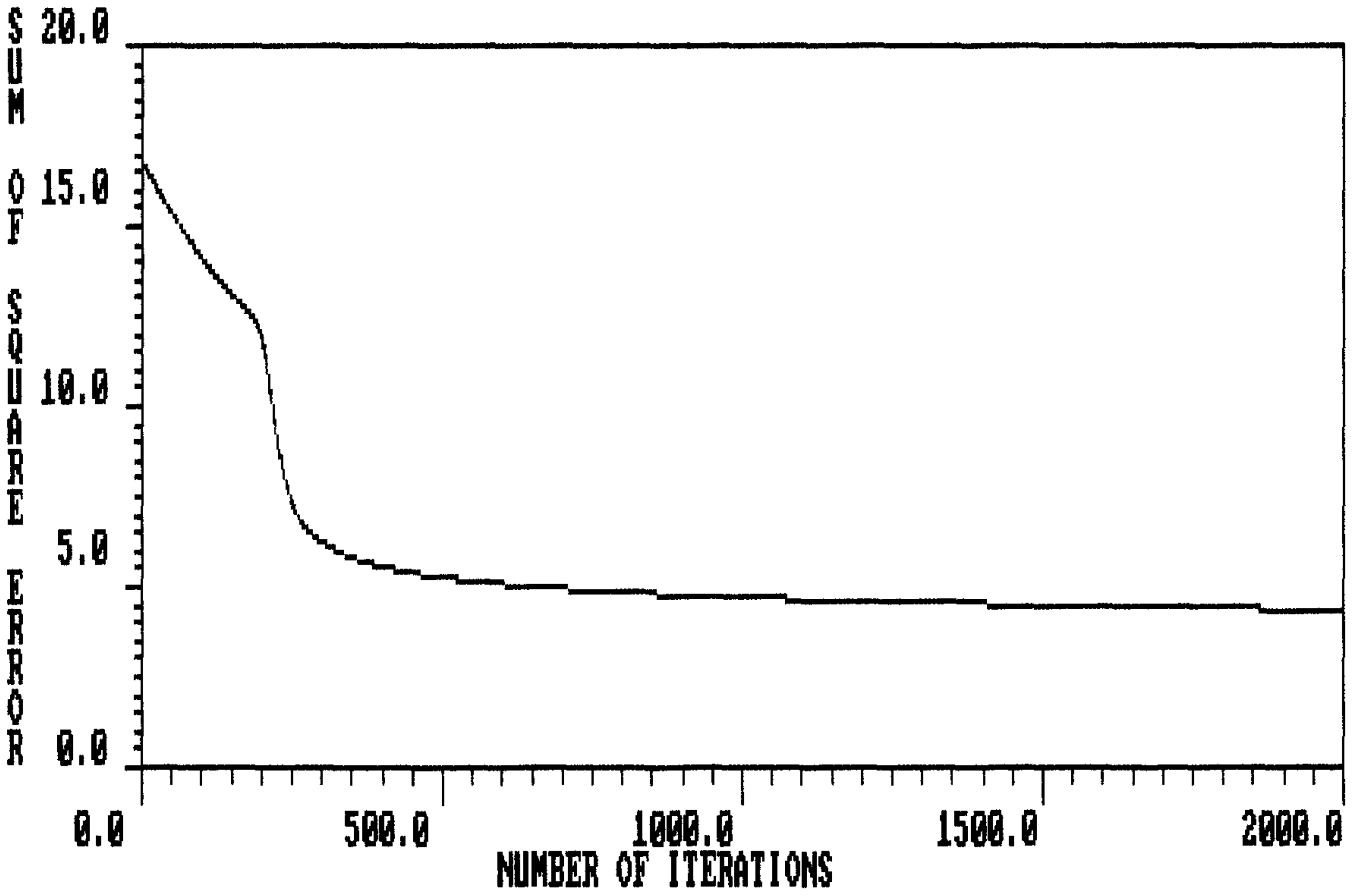


Fig. 6.4 Neural Network Learning Curve
Network Topology : 20-62-1 Learning Algorithm : BP(Exp. Smoothing)

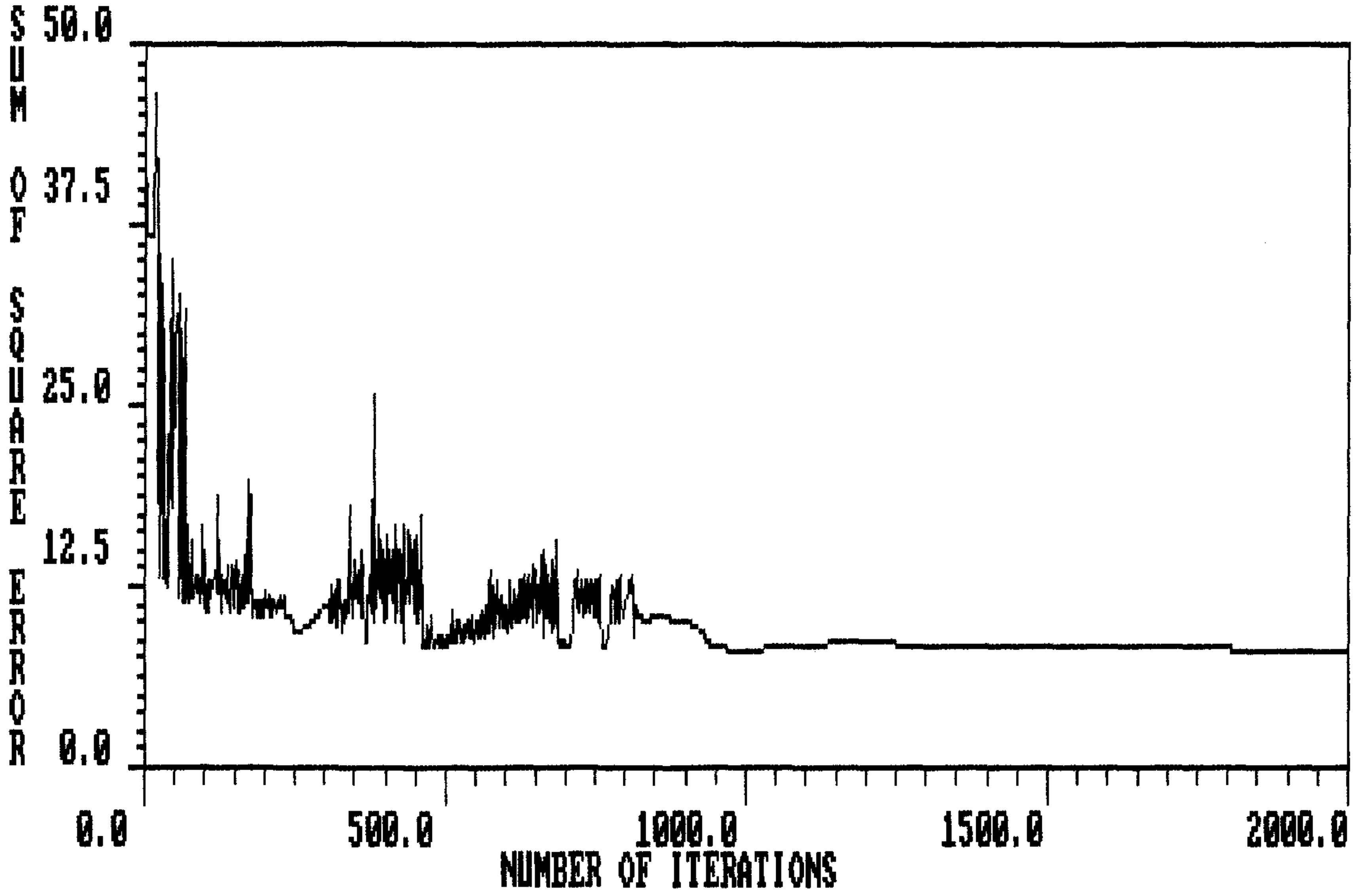


Fig. 6.5 Neural Network Learning Curve
Network Topology : 20-11-2 Learning Algorithm : BP(Momentum)

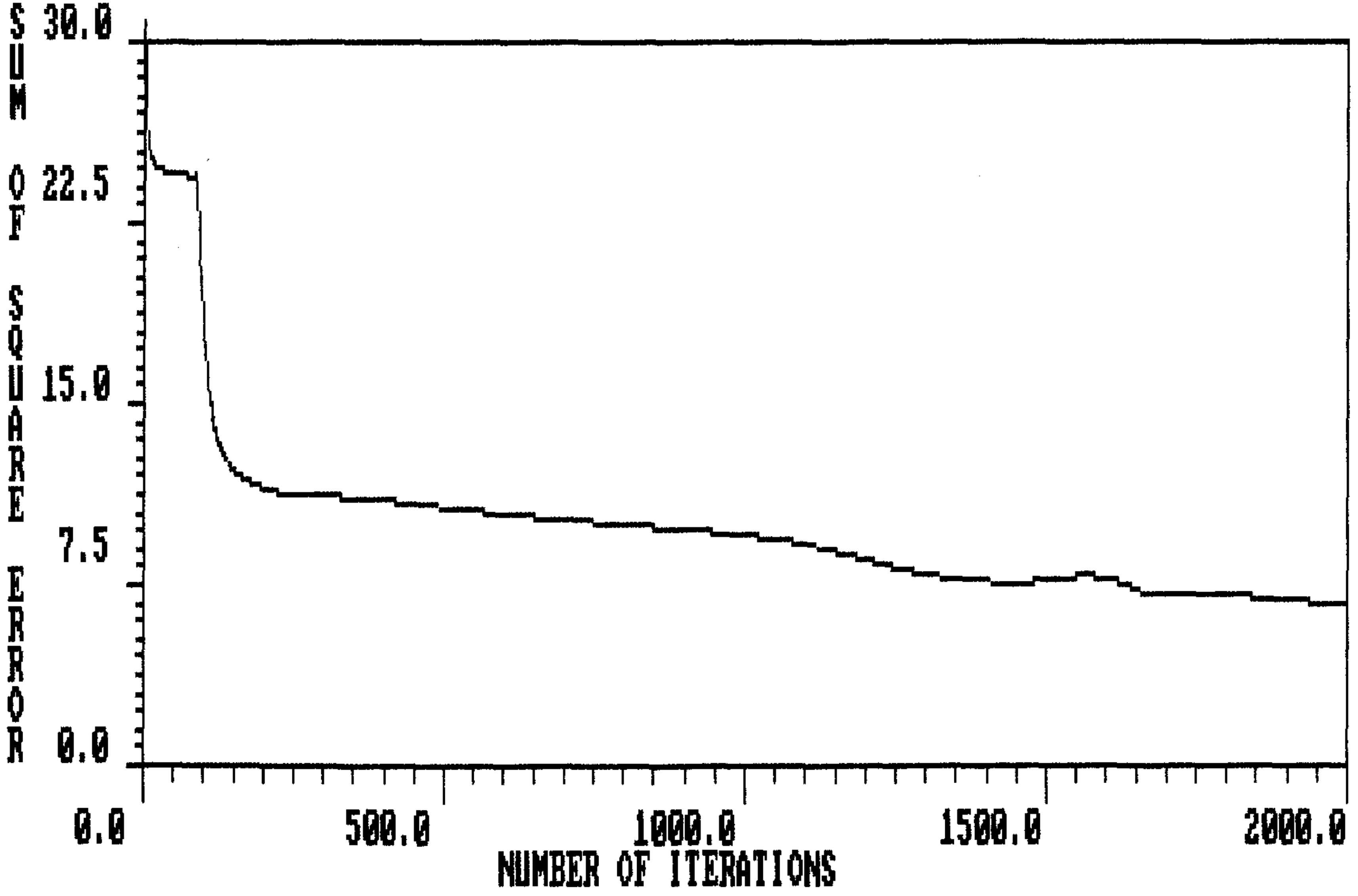


Fig. 6.6 Neural Network Learning Curve
Network Topology : 20-11-2 Learning Algorithm : BP(Exp. Smoothing)

TABLE 6.1

RESULTS OF SIMULATION

Network Topology	Learning Algorithm	Number of iterations	CPU time (in secs.)	Minimum Sum of square error	Correct classification (%)	Misses (%)	False Alarm (%)	Classification performance grading
20-62-2	BP (mom)	1000	10687	8.07884	60.5	17.3	22.2	1
20-62-2	BP (exp. smoothing)	2000	26000	6.7763	55.3	15.7	29.0	6
20-62-1	BP (mom)	2000	20889	4.27692	58.9	15.7	25.4	2
20-62-1	BP (exp. smoothing)	2000	24974	4.33416	55.7	17.3	27.0	5
20-11-2	BP (mom)	2000	8041	8.03507	57.7	16.1	26.2	3
20-11-2	BP (exp. smoothing)	2000	9318	6.59165	56.5	15.7	27.8	4

from the learning curves.

(d) Regardless of the network topology, the learning is totally jitter-free in exponential smoothing compared to that in the momentum method.

(e) As is evident from the learning curves, for a given network topology, exponential smoothing takes more time to converge than the momentum method.

(f) Momentum method gives better classification result than exponential smoothing.

(g) For a given number of neurons in the input layer, the classification accuracy improves with the number of neurons in the hidden and the output layer. But this is at the cost of computation and network convergence time.

(h) The minimum value of the sum of square error, which corresponds to the generalized weights and biases of the converged network, need not necessarily be a pointer to its classification accuracy.

6.7 Conclusion

The efficacy of using a neural network for target detection is well inferable from the results of the simulation which was presented in this chapter. Some hardware aspects of neural networks and a few proposals for exploiting them for sonar applications are discussed in the next chapter.

CHAPTER 7

THE NEURAL-BASED APPROACH - A RETROSPECT

A pattern recognition application (viz. detection of an underwater target) of a multi-layered feed-forward neural network was discussed in the previous chapters. It was mainly aimed as a feasibility study where the network was simulated by software on a computer. This being an off-line process, a very limited set of data recordings that were available had to be relied upon for generating both the training and the testing patterns. The compression of data due to the preprocessing method adopted here led to further reduction in the number of these generated patterns.

The efficacy of these types of applications can be conclusively established only through extensive evaluation of the actual neural network hardware with large amounts of practical real-time data. Since a neural network is as intelligent as the way it is trained, elaborate studies involving large varieties of underwater targets against different conditions of the sea background are highly scopeful and all the more relevant.

7.1 Neural Networks Perspectives and Potentials

The ever-increasing technological demands of the present-day world require innovative approaches to highly challenging problems. Artificial neural networks with their outstanding features like massive parallelism, ability to learn, association, generalization, flexibility (plasticity), error tolerance etc. offer the promise of better solutions at least to some of these problems. The unusual and stimulating interdisciplinary nature of neurocomputing spans over neuro-sciences, cognitive sciences, psychology, computer science, electronics, physics and mathematics. It has already made its impact in the commercial, industrial, medical and scientific fields. The potential defence applications of neural networks include automatic target recognition (ATR), sonar and radar signal processing, vision and image processing, photonics, artificial intelligence, robotics, expert systems etc. .

7.2 Sonar Applications of Neural Networks - Some Proposals

Nowadays, the science of anti-submarine warfare (ASW) is emerging as a highly competitive field. With the present-day technology, such integrated and sophisticated defensive measures like sonar systems (hull-mounted, towed array, variable depth and helicopter-mounted types), weapon controls and a variety of electronic and acoustic countermeasures come to the rescue of surface ships. Even then, highly manoeuvrable submarines, which are made more silent with nuclear propulsion and ingenious design technology and which are equipped with computer-controlled wire-guided torpedoes and long-range nuclear missiles, do reign the sea with terror. In the wide open ocean, ultimate success inevitably goes to the one who detects the enemy first. Hence, the earliest detection and quickest localization of the foe are premier factors and in this endeavour, neural networks are capable of playing their vital role.

7.2.1 Sensor Failure Detection

A sonar system inevitably includes a hydrophone array at its wet end. It is used to transform the hydroacoustic signals into electrical signals which are further processed for detecting the target. The spatially distributed elements (sensors) of this array, by virtue of their arbitrary geometry and dimensionality, spatially discriminates the desired signal against noise and reverberation thereby enhancing the SNR. This process is called beamforming, the effectiveness of which is directly influenced by the width of the beam.

The pattern of spatial distribution of the elements in the array has a direct bearing on the beam width of the array. The failure of an element in the array adversely affects the beam pattern; more the number of failed elements, deeper the extent of this degradation. A neural network may be used for the detection of sensor failure. Signals from all the elements, after appropriate combination and preprocessing, can be fed to a neural network that is already trained with a supervised learning algorithm. The diagnostic output from the network can then be used for either manual or automatic initiation of appropriate remedial measures.

7.2.2 Beamforming

The spatial filtering is accomplished in the beamformer through a series of operations involving the weighting, delay, and summation of the signals received by the spatial elements. The summed-up output is further processed for frequency and temporal discrimination. A time-delay neural network (TDNN), therefore, can directly implement a beamformer. A TDNN has the ability to relate and compare current input to the past history of events. This enables the network to discover acoustic features and the temporal relationships between them independent of position in time so that they are not blurred by temporal shifts in the input [41].

7.2.3 Signal Enhancement

Another area where neural networks may be prospectively employed is in sonar signal enhancement for detection of signals submerged in background noise. The underlying principle upon which neural networks operate being one of pattern recognition, they may be effectively employed for SNR enhancement. Implementation of some of the existing algorithms for signal enhancement through neural network hardware might be practically feasible and worth attempting. Statistical methods, which are more accurate than backpropagation, can be used here for the unsupervised training of the network.

7.2.4 Non-Acoustic Methods of Submarine Detection

Acoustic methods of submarine detection, though widely used and are effective, can be adversely affected by topical conditions. Due to complexities of underwater sound propagation, the acoustic signals are highly susceptible to masking effects.

Non-acoustic methods attempt to detect a submarine by sensing the perturbations it creats in the surrounding physical environment. These disturbances, however, must be measurable and separable from the background of similar naturally occuring disturbances. Phenomena such as wakes, internal waves & disturbances and magnetic & thermal anomalies which are generated by a moving submarine are the important useful parameters in this regard. The perturbations in these being very feeble, devices like superconducting quantum interfacing device (SQUID) are used for measuring them.

Since the characteristics of the submarine and the pattern of perturbations it creats in the vicinity can be mutually correlated, any detection algorithm has to search for these expected signature patterns. A complex neural network stored with all such signatures pertaining to different classes of submarines can carry out a nearest-neighbour search in the sensor data presented to it. These patterns have to be essentially made invariant to the ambient pattern of the sensed parameter and the speed of the searching platform. Optical neural networks, with their inherently high speed and the potential for massive interconnectivity, stand as the best bet for this application.

7.3 Neural Network Hardware

Neural network models in software generally consist of many very densely interconnected processing elements, each of which performs a simple computation in parallel with its neighbours. These models and the learning algorithms are computationally intensive on general-purpose computers. However, because of the computational simplicity of the basic processing element, neural networks are implemented on special-purpose massively parallel hardware which can vastly outperform implementations on even the most powerful serial computer. The neurocomputer hardware has been an essential ingredient for the development of practical applications of neural network technology.

Only VLSI processor arrays can realize the true computing potential of massively parallel neural networks. This realization follows one of the two approaches : (1) general purpose neurocomputers that consist of programmable processor arrays for emulating a range of neural network models (2) special purpose neurocomputers that are dedicated hardware implementations of a specific neural network model. Any programmable neurocomputer is order-of-magnitude slower than its directly fabricated hardware version which has got very poor generality. In fact, far more dedicated special-purpose neural network hardware is being developed than programmable neural processors [19]. The fabrication technology is broadly classifiable as electronic, optical and electro-optical implementations where the second and third are rapidly outgrowing the first one. Researchers now consider molecular devices, still very much in their infancy, as a new basis of neurocomputers.

Silicon implementation can be considered the first step toward large neurocomputers. While considering the possible architectures for the basis of a neurocomputer, the important design issues are parallelism, performance, flexibility - and their relationship to the silicon area. These issues, which are directly influenced by the node complexity, lead to radically different systems which range from simple traditional RAMs to programmable processors and special-purpose dedicated hardwares [18]. For example, " 80170 ETANN " which is a VLSI electrically trainable artificial neural network developed by Intel Corporation, USA, is the fastest device commercially available as on date [43]. It has 64 neurons with a total of 10,240 programmable analog weights. They perform calculations, known as "connections" simultaneously, resulting in a performance of more than two billion connections per second (A connection is a multiplication-and-sum operation). Newer and more powerful neural network chips are also being developed by this company.

The increased circuit density possible in VLSI makes it most suited for neurocomputer implementation on silicon.

But, the main design problems encountered here are: massive interconnectivity of the processing elements, complex adaptivity for the synaptic weights, realization of learning algorithms, network size & geometry, processing and communication speeds and data representation. The number of cells (ie. neurons and synapses) in a fully interconnected neural network grows phenomenally with the number of neurons. Also, the area required to route connections and to contain the average length of interconnections increases at unacceptable rates as more processing elements are added. Reduction of size of the basic cells even despite a loss of precision, wafer-scale integration and three-dimensional integration are only some of the answers to these problems. For better solutions, the architectural properties of VLSI have to be further explored and exploited thoroughly.

Analog optoelectronic hardware implementation of neural nets, which was first introduced in 1985, has many advanteges over the electronic implementation. Primary among these is that the optoelectronic (photonic) approach combines the best of the two worlds: viz.(i) the massive interconnectivity and parallelism of optics and (ii) the flexibility, high gain and decision-making capability offered by electronics. It seems more attractive to form analog neural hardware by completely optical means where switching of signals from optical to electronic carriers and vice versa is avoided. However, in the absence of suitable fully optical decision-making devices, the capabilities of the optoelectronic approach remain quite attractive and stand competitive with other approaches when considering the flexibility of architectures possible with it [42].

Molecular electronics is a vision that promises to solve all the technological problems of neural networks. Its self-building and self-organizing capabilities in three dimensions offer prospects of huge neural networks occupying compact space and rendering highly superior performance. In reality, however, the development of biological molecular electronics will be very slow and the field of physical molecular electronics is just starting with the first test structures [19].

7.4 Conclusion

Biological neural nets were evolved in nature for one ultimate purpose : that of maintaining and enhancing survivability of the organism they reside in. Embedding artificial neural nets in man-made systems, and in particular autonomous systems, can serve to improve their survivability and hence reliability. Neurocomputers can be expected to play an important role in the modeling and study of highly complex systems and problems with enhanced flexibility and speed offered by integrated optoelectronic techniques.

APPENDIX 1

REFERENCES

[1] D.E.Rumelhart and D.Zipser,"Feature Discovery by Competitive Learning",in D.E.Rumelhart & J.L.McClelland(Eds.),"Parallel Distributed Processing : Explorations in the Microstructure of Cognition, Volume 1 : Foundations", MIT Press, 1986.

[2] D.E.Rumelhart, G.E.Hinton & R.J.Williams,"Learning Internal Representations by Error Propagation", in D.E.Rumelhart and J.L.McClelland(Eds.),"Parallel Distributed Processing : Explorations in the Microstructure of Cognition, Volume 1 : Foundations", MIT Press, 1986.

[3] Geoffrey E. Hinton, "Connectionist Learning Procedure" in "Artificial Intelligence", Elsevier Science Publishers B.V., 1989 , pp. 185 - 234 .

[4] Nick Beard, Antonia Jones, "Harnessing Neural Networks", Electronics World + Wireless World, December 1990, pp.1047 - 1052 .

[5] Dr. P.S.Sastry, Prof. M.A.L.Thathachar and Dr. K.R.Rama-krishnan, "Artificial Neural Networks : A Tutorial Course", IEEE ACE '90 Conference (India), January 1991.

[6] Alexander I and Morton H,"Introduction to Neural Computing", Chapman & Hall, 1990.

[7] Robert Hecht-Nielsen,"Neurocomputing : Picking the Human

Brain", IEEE Spectrum, March 1988, pp. 36 - 41 .

[8] William T.Illingworth,"Beginner's Guide to Neural Networks", IEEE AES Magazine, September 1989, pp. 44 - 49 .

[9] Philip D.Wasserman,"Neural Computing : Theory and Practice", Van Nostrand Reinhold, New York, 1989.

[10] G. Venkataraman and G. Athithan,"Spin Glass, the Travelling Salesman Problem, Neural Networks and all that", Pramana - Journal of Physics, Vol.36, No.1, January 1991, pp. 1 - 77 .

[11] Hans P. Graf and Lawrence D. Jackel, "Analog Electronic Neural Network Circuits",IEEE Circuits and Devices Magazine, Vol.5, No.4, July 1989, pp. 44 - 55 .

[12] Les E. Atlas and Yoshitake Suzuki,"Digital System for Artificial Neural Networks", IEEE Circuits and Devices Magazine, Vol.5, No.6, November 1989, pp. 20 - 24 .

[13] Mary Ann C. Maher et al.,"Implementing Neural Architectures Using Analog VLSI Circuits", IEEE Trans. on Circuits and Systems, Vol.36, No.5, May 1989, pp. 643 - 652 .

[14] David E. Van Den Bout and Thomas K. Miller III,"A Digital Architecture Employing Stochasticism for the Simulation of Hopfield Neural Nets", IEEE Trans. on Circuits and Systems, Vol.36, No.5, May 1989, pp. 732 - 738 .

[15] Mahmoud K. Habib and H. Akel, "A Digital Neuron-Type Processor and Its VLSI Design", IEEE Trans. on Circuits and Systems, Vol.36, No.5, May 1989, pp. 739 - 746 .

[16] Kwabena A. Boahen et al.,"A Heteroassociative Memory Using Current Mode MOS Analog VLSI Circuits", IEEE Trans. on Circuits and Systems, Vol.36, No.5, May 1989, pp. 747 - 755.

[17] Mark Walker, Paul Hasler and Lex Akers,"A CMOS Neural Network for Pattern Association", IEEE Micro, Vol.9, No.5, October 1989, pp. 68 - 74 .

[18] Philip Treleaven, Marco Pacheco and Marley Vellasco,"VLSI Architectures for Neural Networks", IEEE Micro, Vol.9, No.6, December 1989, pp. 8 - 27 .

[19] Karl Goser et al.,"VLSI Technologies for Artificial Neural Networks",IEEE Micro, Vol.9, No.6, December 1989, pp. 28-44.

[20] Michel Verleysen and Paul G.A. Jespers,"An Analog VLSI Implementation of Hopfield's Neural Network", IEEE Micro, Vol.9, No.6, December 1989, pp. 46 - 55 .

[21] Olivier Rossetto et al.,"Analog VLSI Synaptic Matrices as Building Blocks for Neural Networks ", IEEE Micro, Vol.9, No.6, December 1989, pp. 56 - 63 .

[22] Alan F. Murray,"Pulse Arithmetic in VLSI Neural Networks", IEEE Micro, Vol.9, No.6, December 1989, pp. 64 - 74 .

[23] Paul W. Hollis and John J. Paulos, "Artificial Neural Networks Using MOS Analog Multipliers", IEEE Journal of Solid-State Circuits, Vol.25, No.3, June 1990, pp.849 - 855.

[24] Bernard Widrow et al., "Layered Neural Nets for Pattern Recognition" , IEEE Trans. ASSP, Vol.36, No.7, July 1988,

pp. 1109 - 1117 .

[25] R. Paul Gorman and Terrence J. Sejnowski , "Learned Classification of Sonar Targets Using a Massively Parallel Network", IEEE Trans. ASSP, Vol.36, No.7, July 1988, pp. 1135 - 1140 .

[26] Richard P. Lippmann, "Neural Nets for Computing", 1988 Int'l Conf. on ASSP, Vol.1 (Speech Processing), pp. 1 - 6 .

[27] Panos J. Antsaklis, "Neural Networks in Control Systems", IEEE Control Systems Magazine, Vol.10, No.3, April 1990, pp. 3 - 5 .

[28] Richard P. Lippmann, "An Introduction to Computing with Neural Nets", IEEE ASSP Magazine, April 1987, pp. 4 - 22 .

[29] Robert J. Urick,"Principles of Underwater Sound", McGraw-Hill Book Co:, New York, 1975.

[30] Alan A. Winder, "Sonar System Technology", IEEE Trans. on Sonics and Ultrasonics, Vol. SU-22, No.5, September 1975, pp. 291 - 332 .

[31] Michael W. Roth,"Survey of Neural Network Technology for Automatic Target Recognition",IEEE Trans. on Neural Networks, Vol.1, No.1, March 1990, pp. 28 - 43 .

[32] D.O.Hebb,"The Organization of Behaviour", John Wiley & Sons, New York, 1949.

[33] Frank Rosenblatt, "Principles of Neurodynamics", Spartan

Books, New York, 1962.

[34] W.S.McCulloch and W.Pitts, "A Logical Calculus of the Ideas Immanent in Nervous Activity", Bulletin of Mathematical Biophysics, Vol.5, 1943, pp. 115 - 133 .

[35] R.O.Duda and P.E.Hart, "Pattern Classification and Scene Analysis", John Wiley & Sons, New York, 1973 .

[36] B.Widrow and S.D.Stearns, "Adaptive Signal Processing", Prentice-Hall, New Jersey, 1985 .

[37] G.G.Lorentz, "The 13th Problem of Hilbert" in F.E.Browder (Ed.),"Mathematical Developments Arising from Hilbert Problems", American Mathematical Society, Providence, RI, 1976 .

[38] P. Werbos,"Beyond regression : New tools for prediction and analysis in the behavioral sciences", Ph. D. dissertation, Harvard University, Cambridge, MA, August 1974 .

[39] D.B.Parker, "Learning logic", Tech. Rep. TR-47, Center for Comput. Res. Econ. and Manage. Sci., Mass. Inst. of Technol. Cambridge, April 1985 .

[40] B.Bhanu, "Automatic target recognition : State of the art survey",IEEE Trans. Aerospace Electron. Syst. , Vol. AES-22, No.4, July 1986, pp. 364 - 379 .

[41] Alexander Waibel et al. , "Phoneme Recognition Using Time-Delay Neural Networks",IEEE Trans. ASSP, Vol.37, No.3, March 1989, pp. 328 - 339 .

[42] Nabil H. Farhat,"Optoelectronic Neural Networks and Learning Machines",IEEE Circuits and Devices Magazine, Vol.5, No.5, September 1989, pp. 32 - 41 .

[43] Microcomputer Solutions (A Publication of Intel Corporation, California, USA), March / April 1991, pp. 4 - 6 .