Enhancing the Applied Epistemics of an Expert System for Management Applications Using Neural Networks

Thesis Submitted for
the Award of the Degree of

Doctor of Philosophy

Brij Bhushan Sahni

Department of Computer/Information Sciences
School of Mathematics & Computer/Information Sciences
University of Hyderabad

1994

Department of **Computer/Information** Sciences
School of Mathematics & Computer/Information Sciences
University of Hyderabad

Certificate

This is to certify that the thesis entitled **Enhancing Applied Epistemics of an Expert System for Management Applications using Neural Networks** being submitted by **Brij Bhushan Sahni** for the fulfillment of the requirements for the award of the degree of Doctor of Philosophy in Computer Science is a record of bonafide work carried out by him under our supervision at the University of Hyderabad.

The matter embodied in this thesis has not been submitted for the award of any other degree.

[illegible]

Prof. P.G. Reddy
Supervisor

[illegible]

Dr. Arun Agarwal
Supervisor

[illegible]

Prof. A.K. Pujari
Head
Department of Computer/
Information Sciences

[illegible]

Prof. V. Kannan
Dean
School of Mathematics &
Computer Science

Acknowledgement

I am overwhelmed with gratitude to acknowledge the support and encouragement that I received during the course of this work.

I express my respectful regards and gratitude to my research supervisor Prof. P.G. Reddy for his scholarly guidance, constructive criticism and constant encouragement throughout. I have been immensely benefitted by his scholarship and inspired by his dedication to work. I shall always cherish my association with him.

I feel words are inadequate to convey my feelings of eternal gratitude to Dr. Arun Agarwal. I will remain indebted to Dr. Agarwal for introducing me to the fascinating world of AI and for patiently initiating me and guiding me throughout. I feel I have been extremely lucKy to have a respected scholar like him as my guide.

I wish to pur on record my deep sense of gratitude to Prof- E.A. Feigenbaum for sparing time for me and for his encouragement-

I am grateful to Prof. A.K. Pujari, Head, Department of Computer/Information Sciences for constant encouragement and support.

I am indebted to the members of faculty of Computer Sciences, especially to Prof. B.E. Prasad, Dr. A.S. Reddy, Dr. Uma, Dr. K.C. Reddy, Dr. P.N. Girija, Dr. P.S. Rao and Dr. P.R.K. Murthy, who helped me. My sincere thanks are also due to the Dean, Prof. V. Kannan, as well as to Prof. M. Sitaramayya for their help and encouragement during my tenure as a Research Scholar of the School of MCIS.

My special thanks are due to Mr. Atul Negi and Mr. H.P. Chattopadhyay for their total support.

I also thank Mr. Kameshwar Rao and Mr. Balakrishna of AI Lab for their timely help.

My thanks are due to Ms. Jayasri, Mr. Subrahmanyam, Mr. Murthy, Ms. Bharati, Ms. Rohini and other staff of AI Lab and School of Mathematics & Computer/Information Sciences for their support.

I gratefully acknowledge the timely support received from the University authorities in the completion of this work.

My special thanks are due to my friends Mr. Satyanarayana and Mr. Narasimha Rao for their help .

I should'nt forget to acknowledge my main and constant source of inspiration, support and encouragement—my wife, without her constant support I would not have completed this work.

Finally, I thank all the people who have not been individually acknowledged but have contributed in some way or other in the completion of this work.

— Brij Bhushan Sahni

Abstract

Neural Networks (NN) are the latest additions to the field of Artificial Intelligence (AI). They are now being applied to improve the performance of conventional rule-based expert systems (CES). In the present work a system that can overcome many of the limitations of CES is presented. This system enhances the applied epistemics and is ideal for developing expert systems for management applications. The system aims at achieving high reliability through major innovations that avoid the traditional bottlenecks. Neural network (NN), especially knowledge-based artificial neural network, is integrated into the CES. This combination results in a high reliability level and a high throughput that are essential for management applications. In management applications such as applications for share market it is vital that the expert system gives outputs that are as close to the opinion of an expert. That is the certainty factor given by the expert and the system should be very close. To achieve this kind of accuracy we have specially developed a neural network (NN) based framework. Due to this framework, non-binary rules from the rule base can be napped onto a network. The NN that works on back-propagation methodology has been substantially modified and integrated into an expert system specially built to bring about a fine combination of NN and conventional expert system technology. An incisive analysis of equity shares is presented from a neuro-knowledge engineer's perspective. A rule-base has been formulated on cement industry. The NN effectively checks the consistency of the rules and enhances the reliability to map the rules. A special user interface has been built to enable even a management professional who is not a computer scientist to interact effectively with the system. A training methodology is developed for NN to get the required level of certainty factor for the goal state.

Contents

1 Introduction, Motivation, Problems, Perspective l

1.1 Introduction 1

1.2 Expert Systems 3

1.3 The Present Work 10

1.4 Organization of the Thesis 12

2 Neural Computing—History and Current Trends 15

2.1 Historical Review 15

2.2 The Biological Neurons 17

2.3 The Artificial Neurons 13

2.4 Artificial Neural Networks 20

2.5 Network Paradigms 21

2.6 Applications of Neural Computers 50

3 Expert System and Knowledge-based Artificial Neural Network 55

3.1 Mapping Rule-based Systems into Neural Architecture 57

3.2 Knowledge Representation 60

3.3 Inference 63

3.4 Learning 66

3.5 Tuning a Rule-base Using Neural Nets 75

3.6 Inducing Rules for a Connectionisr ES 78

3.7 Present Work 80

4 Handling Uncertainty Using ANN for Non-binary Inputs 82

4.1 Reasoning Under Uncertainty 32
4.2 Building Blocks to Implement Non-binary Values 87
4.3 Representation of Rule Format, Parsing and Rule-base 98
4.4 Evidential Reasoning: NN Approach 100
4.5 Training/Learning Factual Information 103
4.6 Rule-base Consistency 104

5 Equity Shares—An Analysis: A Neuro Knowledge Engineers Perspective 106

5.1 Equity Shares 106
5.2 Analysis of Equity Investment 107
5.3 Technical Analysis 114
5.4 Advantages of Equity Investment 116
5.5 A Framework for Analysis and Decisions 119
5.6 An Example 123

6 Inside Neuro Expert: Implementation and Results 129

6.1 System Design 130
6.2 Knowledge Representation 148
6.3 User Interface and Consult Facility 165

7 Summary and Future Directions 180

References 186

Chapter 1

Introduction, Motivation, Problems, Perspective

1.1 Introduction

Success of any organization depends upon dynamic and apt decisions. The decision-making process is a complex process involving numerous parameters. If the success of any organization is analyzed, it will be noted that the organization has been able to reach this peak because of certain timely and relevant decisions made by its leaders. This crucial leadership is provided by the managers.

Managers have traditionally performed better than computers in activities that involve decision-making which is an act of intelligence. However, managers have successfully utilized computers to enhance their decision- making capabilities. Managers initially utilized computers by automating the task of record keeping. Then came the database management systems, decision support systems and finally the principle of a generalized problem-processing system (GPPS) [Holsapple and Whinston 1989] emerged. All these systems helped the manager in storing, retrieving, manipulating, and collating data. The need for intelligent systems that could further enhance the decision making capabilities of a manager and help him to achieve the following was still strongly felt:

(a) To respond very flexibly to situations.

(b) To make sense out of ambiguous or contradictory messages.

(c) To recognize the relative importance of different elements of a situation.

(d) To find dissimilarities between situations despite similarities which may link them.

The above tasks require in varying degrees, intelligent computers, if they have to be productive tools in the hands of a manager. Efforts to make computers intelligent gave birth to a field called Artificial Intelligence (AI).

The field of Artificial Intelligence provides relevant techniques and methodologies to meet the goals mentioned above.

Barr and Feigenbaum [1982] aptly defined AI in the following manner.

> "Artificial intelligence is the part of computer science concerned with designing intelligent computer systems, that is, computer systems that exhibit the characteristics we associate with intelligence in human behavior".

The principal areas of AI research are many, but Expert System (ES) is discussed in the next section as this is closest to the goals mentioned above and this thesis focuses on the applied epistemics of an expert system for such managerial decision-making. There is abundant literature available on expert systems. We briefly discuss the concept of expert system and review its limitations.

1.2 Expert Systems

Currently, the most well known area of AI research is expert systems. Concepts from AI when applied to solve specialized problems came to be known as expert systems. These contain both declarative knowledge and procedural knowledge to emulate the reasoning process of human experts [Mishkoff 1985].

Buchanan and Shortliffe [1984] suggest that a good expert system must be useful, usable, educational when appropriate and able explain its advice, respond to simple questions, learn new knowledge and easily modified.

Therefore, we infer that a good expert system should provide the following:

(a) Expertise when a human expert is not available.

(b) Expertise more uniformly and rapidly available than from human experts.

(c) Assistance to the expert in making decisions that involve many interacting complex factors.

(d) A repository for currently undocumented expert knowledge and procedure, and

(e) A common repository for a dynamically growing knowledge base.

A sample of some representative existing expert systems and problem areas they address are listed in Table 1.

Table l. Some representative expert systems.

Expert system	Problem area
MACSYMA [Martin and Fateman 71]	Solves differential and integral calculus problems in applied mathematics.
INTERNIST, CADUCEUS [Pople 75]	Diagnoses internal medical ailments.
MYCIN [Shortliffe 76]	Diagnoses and prescribes treatments for bacterial blood infections.
CASNET [Weiss et al. 78]	Diagnoses glaucoma and recommends therapies.
PUFF [Kunz 78]	Diagnoses lung disfunctions.
SACON [Bennett and Englemore 79]	Advises how to analyze mechanical structures.
PROSPECTOR [Duda et al. 79]	Determines the major types of ore deposits present in a geological site.
CRYSALIS [Englemore and Terry 79]	Determines the protein structures of unidentified molecules from electron density maps.
DENDRAL [Lindsay 80]	Determines the chemical structures of unidentified molecules.
Rl, XCON [McDermott 81]	Determines an appropriate computer system configuration for customer's needs.

Some of the applications of expert systems as given by Hayes-Roth et al. [1983] are presented in Table 2.

Table *2.* Some applications of expert systems.

Category	Problem addressed	Types of systems
Interpretation	Infers situation descriptions	Speech understanding, image analysis, surveillance
Prediction	Infers likely consequences of given situations	Weather forecasting, crop estimation
Diagnosis	Infers system malfunctions from observations	Medical, electronic
Design	Configures objects under constraints	Circuit layout, budgeting
Planning	Designs actions	Automatic programming, military planning
Debugging	Prescribes remedies for malfunctions	Computer software
Repair	Executes a plan to administer a prescribed remedy	Automobile, computer
Instruction	Diagnoses, debugs, and corrects student behavior	Tutorial remedial
Control	Interprets, predicts, repairs and monitors behaviors	Air traffic control, battle management

It is evident from Tables 1 and 2 that there is a lack of expert systems on management applications such as equity shares related decisions, planning, analysis, etc.

The following are the most important components necessary for building an expert system:

(a) **Domain expert:** Is an individual who has significant expertise in the domain of the expert system being developed.

(b) **Knowledge engineer:** Is usually an AI specialist who works with the domain expert to encode the knowledge.

(c) **Knowledge base:** It is an essential component of all expert systems. It contains the formal representation of the knowledge provided by the domain expert as encoded by the knowledge engineer. Knowledge is encoded using various knowledge representation methods, such **as,** frames, production rules, semantic nets, scripts, conceptual dependencies, etc.

A knowledge base contains both declarative knowledge (facts about objects, events and situations) and procedural knowledge (Information about courses of action). The most popular knowledge representation technique used in expert systems is the rule-based production system.

(d) **Inference engine:** It is responsible **for** interpreting the contents of the knowledge base in the context of a user specified input or hypothesis to reach a goal or conclusion. The inference engine can be divided **into** three parts

i) **Context block:** This part contains the current state of the problem and the solution.

ii) **Inference reasoning mechanism:** This part searches the appropriate set of knowledge and data with the help of the context block to reach a goal or a conclusion.

iii) **Explanation facility:** This facility helps the user to understand the line of reasoning of the expert system.

(e) **Knowledge acquisition facility:** This is to facilitate the assimilation of new knowledge with the present knowledge base.

(f) **User interface:** This component of the expert system permits the user to interact with the expert system with ease.

1.2.1 Conventional Rule-based Expert System

In a rule-based system, the procedural knowledge is integrated in the form of heuristic "if then" rules with the declarative knowledge. However, it is not necessary that all rules would pertain to the domain of the system; some production rules, called meta-rules help guide the execution of an expert system by determining under what conditions certain rules should be considered instead of other rules.

Most of the expert systems developed so far have production rules as their knowledge representation schemes and are termed as rule-based expert system (Fig. 1). Henceforth rule-based expert system is refered to as conventional expert system (CES).

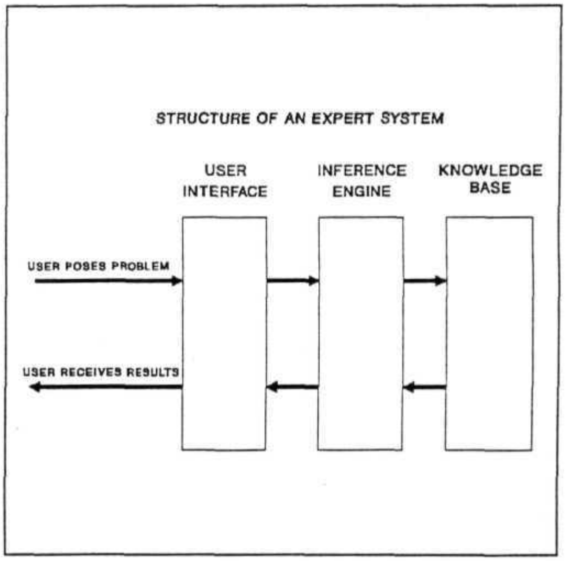


Figure 1. Structure of an expert system.

The salient features of CES are:

(a) *A CES incorporates practical human knowledge expressed in terms of conditional if-then rules.*

(b) *A CES solves a wide range of possibly complex problems by selecting relevant rules and combining their result in appropriate ways.*

(c) The CES determines dynamically the best rules to execute.

(d) A CES explains its conclusions by retracting its actual line of reasoning and explaining the logic of each rule employed.

The knowledge base of CES consists of rules (production memory) and facts (working memory). Rules always express a condition, with an antecedent and a consequent component. The interpretation of a rule is that if the antecedent condition can be satisfied, the consequent also can be satisfied. When the consequent defines an action, the effect of satisfying the antecedent is to schedule the action for execution. When the consequent defines a conclusion, the effect is to infer the conclusion.

The inference engine consists of a recognition-act cycle that has three phases: Match phase, select phase and execute phase. In match phase, rules that satisfy the left hand side of the working memory are selected. These rules are then entered in the conflict set. Select phase selects one rule amongst those in conflict set depending on some criteria. Execute phase fires the selected rule and changes working memory elements accordingly.

Much of knowledge which humans reason with, is inadequate in some respect or other. Sometimes a problem will necessitate probabilistic assessment of decision. In addition, a knowledge engineer may wish to attach confidence measurements (or lack of confidence) for both hypotheses and conclusions. Problems such as these require that the expert system be capable of dealing with knowledge having varying grades of certainty. Because the world does not behave in a strictly Bayesian or Stochastic fashion, several expert systems exploit decision theory to supplement the inference process.

Various theories have been developed to accommodate such uncertainty. These will be discussed in detail in Chapter 4.

1.2.2 The Limitations of CES

In CES, knowledge is expressed as quanta (packets of knowledge) of knowledge thrown into the system without any relation to each other. Inferencing is used to establish interaction. This leads to several problems, they are:

(a) Rules can interact only through working memory. This is a major bottleneck if there are numerous rules to be processed.

(b) In a situation where multiple hypotheses are to be dealt with, forward chaining method is applied. In such cases the CES can rarely explore all the alternatives from among the available multiple hypotheses. It is desirable that search be continued to explore all alternatives so that these can be ranked and the best possible hypotheses be selected.

(c) The refinement of a rule base is a difficult problem (Valtorta 1989; Wilkins and Buchanan 1986) . Refinement is done by interaction with human experts (Eshelman and McDermott 1986; Kahn et al. 1985), machine learning (Michalski et al. 1983; Quinlan 1987), explanation based on domain theory (Mitchell et al. 1986; Smith et al. 1985) and empirical refinement.

    These approaches are expensive and involve a lot of time and effort minimizing the involvement of experts again and again can substantially reduce the cost of maintaining expert system (Fu 1991)

(d) A CES primarily conducts symbolic reasoning but uncertainty is often handled numerically (Fu 1991).

(e) The CES using uncertainty paradigm does not have systematic methodology for debugging a given factual information to assert a conclusion with the desired certainty factor given by an expert.

    In management applications it is of utmost importance that the certainty factors or the level of confidence of an expert be directly and reliably reflected in the output of the expert system.

1.3 The Present Work

In the present work a system that can overcome many of the limitations of CES is presented. This system enhances the applied epistemics and is ideal for developing expert systems for management applications. The system aims at achieving high reliability through major innovations that avoid the traditional bottlenecks. Neural network (NN), specially knowledge-based artificial neural network, is integrated into the CES. This combination results in a high reliability level and a high throughput that are essential for management applications.

Because of the good demand of the management applications and to meet performance requirements, their (CES') rigid frame-work has made expert system development for management applications a difficult process. In management applications decisions have to be very fast and the output from the system needs to be as reliable as the expert's output. A typical application is an expert system for share market.

In management applications such as applications for share market it is vital that the expert system gives outputs that are as close to the opinion of an expert. That is the certainty factor given by the expert and the system should be very close.

To achieve this kind of accuracy we have specially developed a neural network (NN) based framework.

Due to this framework, non-binary rules from the rule base can be mapped onto a network. The NN that works on back-propagation methodology [Runelhart et al. 1986] has been substantially modified and integrated into an expert system specially built to bring about a fine combination of NN and conventional expert system technology.

Each concluding premise is mapped into a neural node and antecedent premise is mapped into a connection. Knowledge embedded in such networks, accounts for their faster convergence to a desired stage in the learning phase. The knowledge of NN lies in its connections and associated weights.

Belief values are combined and propagated in NN. Here certainty theory is assumed [Frost 1988]. Weight factors are propagated across the connections between various premises. In CES, rules can interact only through the working memory. This presents a problem when numerous rules are to be processed. By mapping the rule base into a NN, a rule is fired only when its premise node is activated; thus the rule interaction becomes distributed over the networks rather than centralized through the working memory. This significantly improves the system performance.

The NN also effectively checks the consistency of the rules and enhances the reliability to map the rules. A special user interface has been built to enable even a management professional who is not a computer scientist to interact effectively with the system.

A training methodology is developed for NN to get the required level of certainty factor for the goal state.

A Block Diagram of the proposed system is shown in Figure 2. This encapsulates our conceptualization of the frame-work for realizing the objectives of the thesis.

1.4 Organization of the Thesis

Chapter two presents an overview of Neural Networks with particular emphasis on the concepts of the knowledge based Neural Networks.

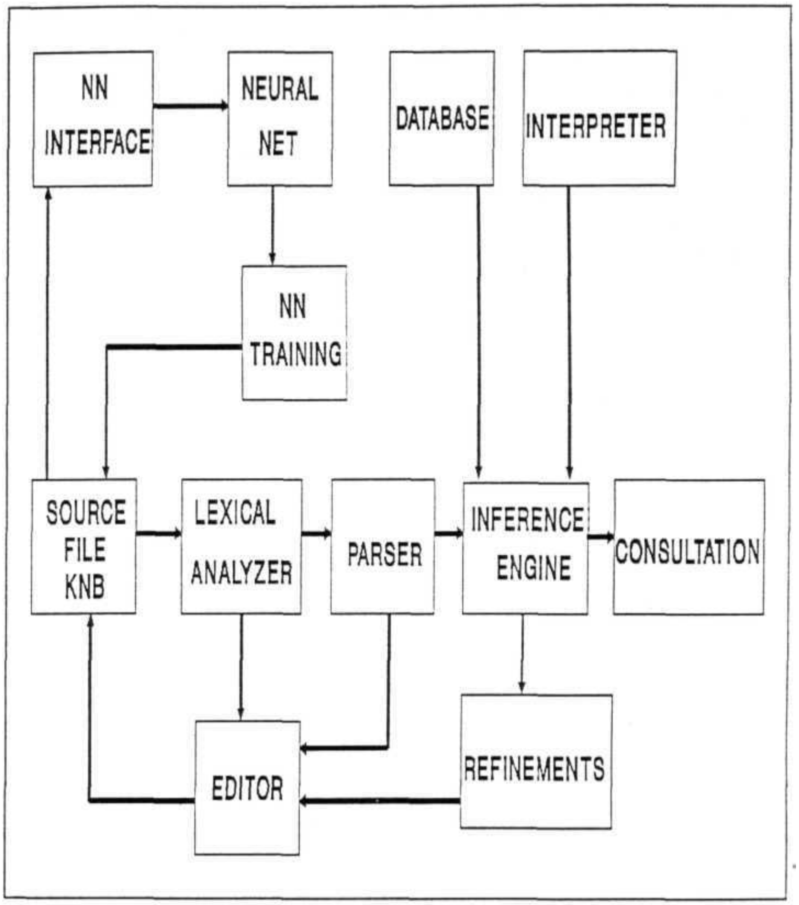


Figure 2. *Block diagram of our proposed model.*

Chapter three presents comparisons and similarities of Expert System and Neural Networks, knowledge representations and inference mechanisms.

Chapter four discusses the concept of tuning a rule base using NN with special emphasis on mapping non-binary inputs in KBNN. It also incorporates a detailed discussion on training the NN and a new approach for handling uncertainties is presented.

Chapter five presents a detailed analysis of equity shares from the perspective of a Neuro knowledge engineer. An analysis of cement industry with sample rules is also presented.

Chapter six presents a detailed discussion on the system we have built. All the implementation aspects are also presented along with the results.

Chapter seven presents the summary of the contributions of this work and suggestions for future extensions.

Chapter 2

Neural Computing—History and Current Trends

Neural computing is the study of networks of adaptable nodes which, through a process of learning from task examples store experimental knowledge and make it available for use [Alexander and Morton 1990]

Neural networks, neural computing, parallel distributed processing, connection science, neuromorphic systems are all synonyms.

2.1 Historical Review

Theoretical explanations of the brain and thinking process were first suggested by some ancient philosophers such as Plato (427-347 B.C.) and Aristotle (384 - 322 B.C.)- Rather physical views of mental processes were held by Descartes (1596-1650).

The class of so-called cybernetic machines to which the neural computer belongs has a much longer history than generally believed. Heron the Alexandrian built hydraulic automata around 100 B.C. William James presented "Enforcing Principle" in 1890. This principle states "When two brain processes are active together or in immediate succession, one of them, on reoccurring tends to propagate its excitement into the other. He also proposed "Non-mathematical description of today's neuron" which states "The amount of activity at any given point in the brain cortex is the sum of the tendencies of all other points that discharge into it, such tendencies being proportionate (l) to the number of times the excitement may have accompanied between such points and that of the point in question (2) to the intensities of such excitement; and (3) to the absence of any rival point functionally disconnected with the first point, into which the discharge might be diverted."

Among the numerous animal models, which have been built to demonstrate need-driven behavior in variable living conditions, one may mention the "protozoan" of Luz from 1920, The "dogs" of Phillips from 1920 to 1930, the "Hemostat" of Ashby from 1948, the "MachinaSpeculatrix" and "Machina Docilis" of Walter from 1950, the "ladybird" of Szeged from 1950, the squirrel from 1951, the "tortoise" of Eichler from 1956, and many versions of a mouse in the "labyrinth" [Nemes 1969].

Abstract, conceptual information processing operations were performed by mechanical devices a couple of centuries ago, for example, the slide rule for the demonstration of syllogisms by Ch. Stanhope [1753-1816], and many mechanisms for set-theoretic and logic operations devised in the 19th century.

Analytical neural modeling has usually, been pursued in connection with psychological theories and neurophysiological research. The first theorists to conceive the fundamentals of neural computing were W.S. McCulloch and W.A. Pitts [1943] from Chicago, who launched this research in the early, 1940s. Models for adaptive stimulus-response relations in random networks were set up by Farley and Clark [1954]. These theories were further elaborated by Rosenblatt [1958]. Widrow and Hoff [1960], Caianiello [1961], and Steinbuch [1961].

Many implementa-tions of neural computers were realized in the 1960s.

2.2 The Biological Neurons

A neuron (Figure 1) can be defined as adaptive node of the brain. There are 10^{11} neurons in an average human brain. Every neuron has a well defined output fibre called the axon. Electrical activity in the shape of short pulses has been observed at the axon which can be said to be firing or not firing. A neuron is said to fire when it amits a buzz of electrical pulses (100 Hz.). The axon usually divides up, forming several endings each of which makes contact with another neuron. The button like terminal where the contact is

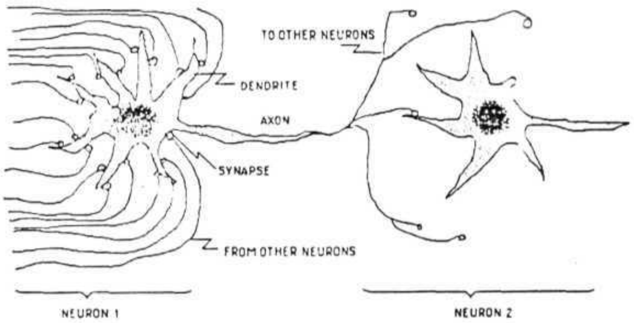


Figure 1. Biological neurons.

Source: Aleksander and Morton 1990

made, is called the synapse. A neuron can receive from 5000 to 15000 inputs from the axon of other neurons. Most neurons have jagged surfaces with carrot like protruburences called dendrites which are the sites for their synapses. Synapses are of two types. When a neuron is aided in firing then that synapse is termed as exhibitory synapse. When a neuron is discouraged from firing, then that synapse is termed as inhibitory synapse.

2.3 The Artificial Neurons

The first formal Artificial neural network model was proposed by McCulloch and Pitts [McCulloch '43] containing sizple two state threshold logic units, with a group of excitory inputs with a group of inhibitory inputs whose action is absolute. The McCulloh Pitts' model formed basis for later models.

In the model (Figure 2) of the neuron (MC? neuron) proposed by w.s. McCulloh and W.A. Pitts, the adaprability

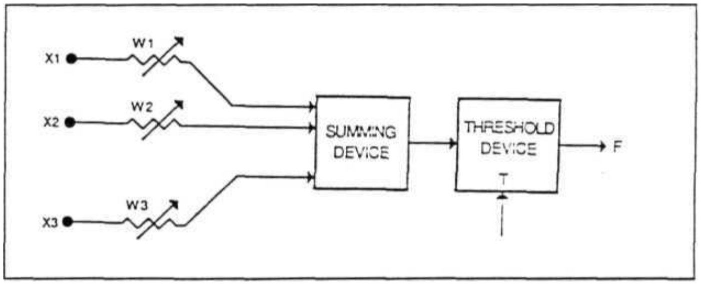


Figure 2. Model of McCulloch Pitts neuron,

of the network comes from representing the intercommunications by a variable weight, which determines the degree to which a neuron should "notice" the net input, it is getting to determine whether it should fire or not. Firing of a neuron is represented as "1" and absence of firing is represented as "0". The effect of an interconnection is represented by weight "W". The effect of interconnection on neuron is then the product X*W, where X is the state of a neuron (i.e., energy state). The fired value for a neuron determines its next state. Negative values of "W" represent inhibitory connections.

In the MCP model, the effects of all neurons connecting a single neuron "i" are added and are passed to some activation in this case a threshold (function) associated with the neuron "i". If the total sum exceeds some threshold T_i (a real number), then the neuron fires.

let $SUM = X_1W_1 + X_2W_2 + \ldots\ldots + X_n W_n$

i.e. $SUM > T_i$ then neuron "i" fires

i.e.,

$$\underset{j}{E}\ X_jW_j > T_j,$$

i.e.,

$$Y = F\ (\underset{j}{\Sigma}\ X_jW_j - T_i)$$

i.e, $Y = 1$ If $\underset{j}{E}\ X_jW_j > T_i$

$Y = 0$ If $\underset{j}{E}\ X_jW_j < T_i$

Some other activation functions are:

- Simple linear function,

  OUT = F(SUM) = K*SUM, where K is some constant

- Squashing function (sigmoid)

  OUT = F(SUM)

  $OUT = 1 / (1 + e^{-SUM})$

the squashing function is also called logistic function. Hence OUT never exceeds some low limits regardless of the values of the NET.

2.4 Artificial Neural Networks

Artificial neural networks (ANN) resemble the brain only superficially. But despite this superficial resemblance, ANN exhibit a surprising number of brain's characteristics. For example, to new ones, and abstract essential characteristics from inputs containing irrelevant data [Wasserman '89]. Hence some of the essential characteristics of artificial neural networks are:

Learning: Learning is a very important characteristic of human brain. It also refers ones ability to respond to novel situations. Artificial neural networks can modify their behavior in response to their environment. Given a set of inputs (perhaps with desired outputs), they self adjust to produce consistent responses. A wide variety of training algorithms have been developed to make the ANN learn. In ANN, learning is done by two ways:

(a) by altering the structure of interconnections between the nodes.

(b) by changing the strengths or signal transnittances i.e., weights of these interconnections.

Generalization: Artificial neural network generalizes automatically as a result of its structure and not by using human intelligence embedded in the form of adhoc computer programs. It is this characteristic which makes the response of the system insensitive to minor variations in its input. Hence it produces a system that can deal with the imperfect world in which we live.

Abstraction: Some artificial neural networks are capable of abstracting the essence of a set of inputs. For example, a network can be trained on a sequence of distorted versions of the letter A. After adequate training , application of another distorted example will cause the network to produce a perfectly formed letter. In some sense, it has learned to produce something that it has never seen before and enumerate.

2.5 Network Paradigms

Although a single neuron can perform certain pattern detection functions, the power of neural computing comes from connecting neurons into networks. The simplest network is a group of neurons arranged in a layer as shown in Figure 3. The circular nodes on the left serve only to distribute the inputs; they perform no computation and hence are not to be considered to constitute a layer. The set of inputs X(input vector) has each of its elements connected to each artificial neuron, shown as squares, through a separate weight. Each neuron simply outputs a sum of the inputs to the network.

Multi-layer networks may be formed simply cascading a group of single layers; the output of one layer provides input to the subsequent layer. Figure 4 shows a Two-Layered neural network.

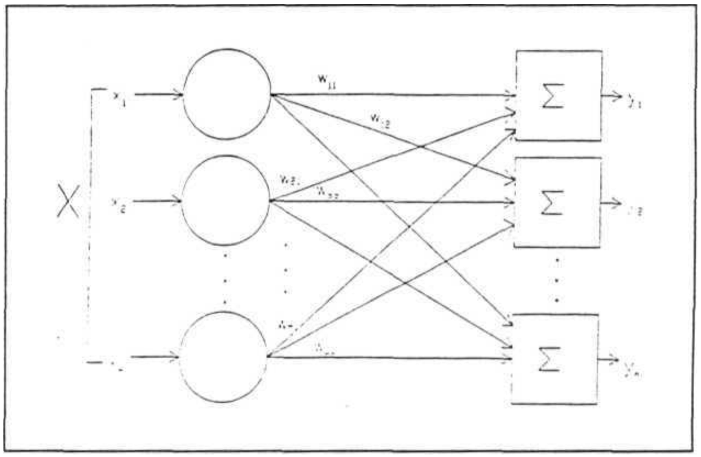


Figure 3. Single layer neural network.

The network described above has no feedback connection i.e., connection through weights extending from the output layers of a layer to the inputs of same or previous layers. Hence they are called concurrent or feed-forward networks. More general networks that contain feedback connections are said to be recurrent. Non-recurrent networks have no memory, their output is solely determined **by the current** inputs and the values of the weights. In some configurations, recurrent networks recirculate previous outputs back to inputs, hence their output is determined both by their current input and their previous outputs.

Outputs may be localized or distributed. We examine some of the popular networks, so that we could exploit their architecture for the thesis.

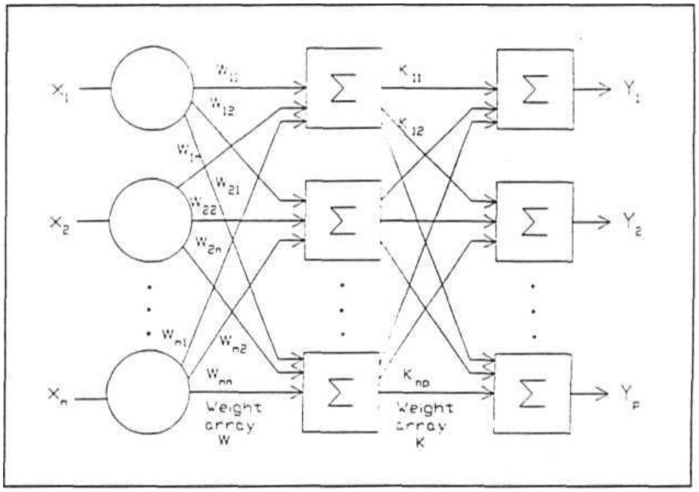


Figure 4. Two-layered neural network.

2.5.1 Supervised Networks: Non-recurrent

2.5.1.1 Perceptrons

In general perceptrons are simply a single-layered network, where in, the neurons of the layer are connected by weights to a set of inputs. This can be used for both continuous valued and binary inputs. In 1960's perceptrons created a great deal of interest and optimism. Rosenblatt [Rosenblatt 89] proved that a perception could learn anything it could represent. The initial euphoria was replaced by disillusionment as perceptrons are found to fail at certain simple learning tasks. Minsky [Minsky 69] proved that there are severe restrictions on what perceptions can learn.

Connection weights and the threshold in a perceptron **can** be fixed or adapted using a number of different algorithms. The original perceptron convergence procedure for adjusting weights was developed by Rosenblatt [Rosenblatt 59].

Linear separability limits single-layer networks to classification problems in which the set of points (corresponding to input values) can be separated geometrically. For a two input case, the separator is a straight line (e.g., Exclusive or function). For three inputs, the separation is performed by a flat plane cutting through creating breakdowns where we must mentally generalize to a space of n dimensions divided by a hyperplane. This limitation of linear separability of single-layer networks can be overcome by adding more layers.

2.5.1.2 Madaline Networks:

Adaptive Linear Neuron (ADALINE)
Many Adaptive Linear Neuron (MADALINE)

In the early 1960s at Stanford, W.C. Ridgeway III initiated an approach to the implementation of non-linearly separable logic functions.

He connected retinal inputs to adaptive neurons in a single layer, and in turn, connected their outputs to a fixed logic device providing the system output.

A typical two-input Adaline is shown in Figure 5.

The following logic device can be simulated as:

AND	W1 = W2 = +1, Xo = +1, Wo = -1.5
OR	W1 = W2 = +1, Xo = +1, Wo = +1.5
MAJ (Majority)	Wl = W2 = W3 = +1, Xo = +1, Wo = 0 (Three input Adaline)

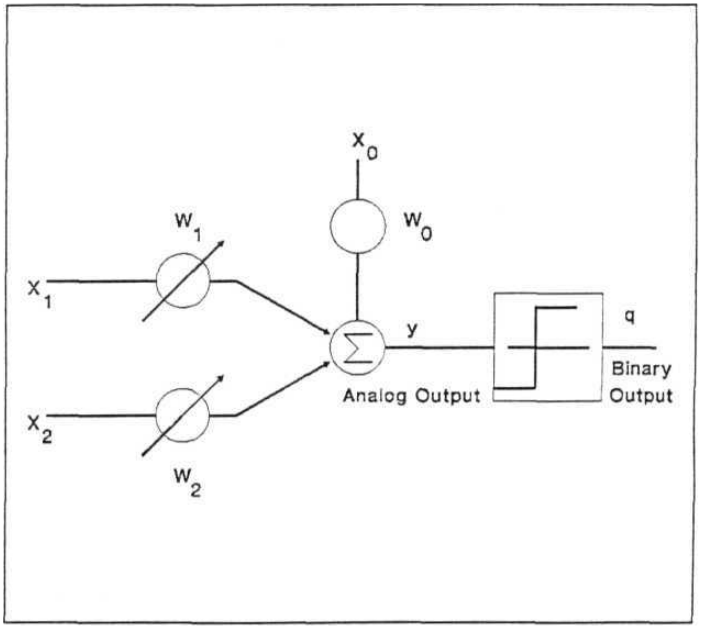


Figure 5. A typical two-input Adaline.

Non-separable functions could thus be solved by neurons in the form of Madaline.

Consider:

X1	**:**	**0**	**0**	**1**	**1**
X2	**:**	**0**	**1**	**0**	**1**
Fl	**:**	**0**	**1**	**1**	**1**
F2	**:**	**1**	**1**	**1**	**0**
F3	**:**	**0**	**1**	**1**	**0**

Fl & F2 are functions simulated by 1st stage of Madaline and F3 simulates "AND" gate in second stage as shown in Figure 6.

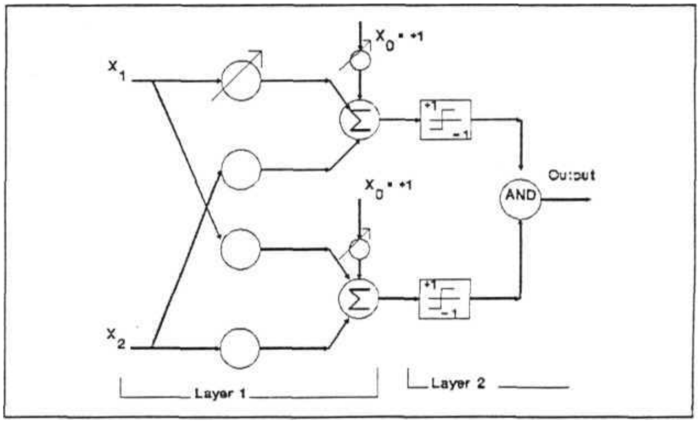


Figure 6. Non-separable functions solved by neurons in Madaline form.

Separating boundaries for the Madaline is shown in Figure 7. Thus "exclusive OR" could be solved by such a two layered network. So linear separability problem could be overcome by adding more layers to a network.

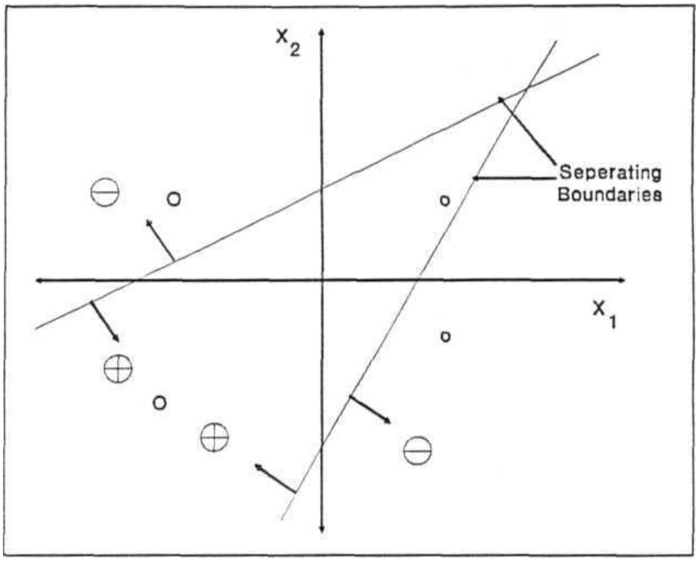


Figure 7. Separating boundaries for the Madaline.

2.5.1.3 Back-propagation

Multi-layer perceptrons are feed-forward nets with one or more layers of nodes between the input and output nodes. These additional layers contain hidden units or nodes that are not directly connected to both the input and output nodes. A three layer perceptron with two layers of hidden units is shown in Figure 8. Multi-layer perceptrons overcome many of the limitations of single layer perceptrons, but were generally not used in the past because effective training algorithms were not available. This has recently changed with the development of new training algorithms [Rumelhart et al.

1986]. Although it cannot be proven that these algorithms converge as with single layer perceptrons, they have been shown to be successful for many problems of interest [Rumelhart et al. 1986].

OUTPUT

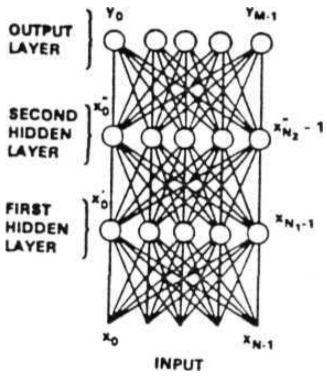


Figure 8. A three-layer perception with two hidden units.

Source: Lippmann 1987

The capabilities of multi-layer perceptrons stem from the non-linearities used within nodes.

As noted above, a single-layer perceptron forms half-plane decision regions. A two-layer perceptron can form any, possibly unbounded, convex region in the space spanned by the inputs.

A three-layer perceptron can form arbitrarily complex decision regions and can separate the meshed classes as shown in the bottom of Figure 9. It can form regions as complex as those formed using mixture distributions and nearest-neighbor classifiers [Duda and Hart 1973]. This can be proven by construction. The proof depends on partitioning the desired decision region into small hypercubes (squares when there are two inputs). Each hypercubes (squares 2N nodes in the first layer (four nodes when there are two inputs) one for each side of the hypercube, and one node in the second layer that takes the logical AND of the outputs from the first-layer nodes. The outputs of second layer nodes will be "high" only for inputs within each hypercube. Hypercubes are assigned to the proper decision regions by connecting the output of each second decision region OR operation in each output node. A logical OR operation will be performed if these connection weights from the second hidden layer to the output layer are

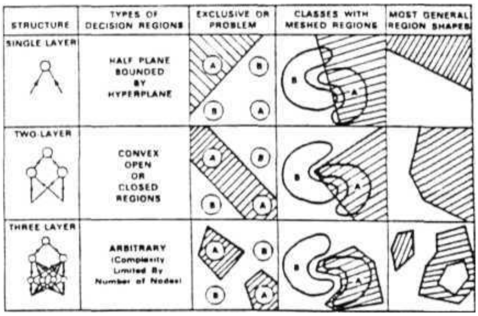

STRUCTURE
TYPES OF DECISION REGIONS
EXCLUSIVE OR PROBLEM
CLASSES WITH MESHED REGIONS
MOST GENERAL REGION SHAPES
SINGLE LAYER
HALF PLANE BOUNDED BY HYPERPLANE
TWO LAYER
CONVEX OPEN OR CLOSED REGIONS
THREE LAYER
ARBITRARY (Complexity Limited By Number of Nodes)
A
B

Figure 9. Types of decision regions that can be formed by single-and multi-layer perceptrons with one and two layers of hidden units and two inputs. Shading denotes decision regions for class A. Smooth and closed contours bound input-distributions for classes A and B. Nodes in all sets use hard limiting non-linearities.

Source: Lippmann 1987

one and thresholds in the output nodes are 0.5. This construction procedure can be generalized to use arbitrarily shaped convex regions instead of small hypercubes and is capable of generating the disconnected and non-convex regions shown at the bottom of Figure 9.

The above discussion centered primarily on multi-layer perceptrons with one output when hard limiting non-linearities are used. Similar behavior is exhibited by multi-layer perceptrons with multiple output nodes when sigmoidal non-linearities are used and the decision rule is to select the class corresponding to the output node with the largest output. The behavior of these nets is more complex because decision regions are typically bounded by smooth curves instead of by straight line segments and analysis is thus more difficult. These nets, however, can be trained with the back-propagation training algorithm [Rumelhart et al. 1986].

The back-propagation algorithm is a generalization of the LMS algorithm. It uses a gradient search technique to minimize a cost function equal to the mean square difference between the desired and the actual net outputs. The desired output of all nodes is typically "low" (0 or < 0.1) unless that node corresponds to the class the current input is from in which case it is "high" (1.0 or > 0.9). The net is trained by initially selecting small random weights and internal thresholds and then presenting all training data repeatedly. Weights are adjusted after every trial using side information specifying the correct class until weights converge and the cost function is reduced to an acceptable value. An essential component of the algorithm is the iterative method that propagates error terms required to adapt weights back from nodes in the output to nodes in the lower layers.

Many researchers have devised improvements and extensions to the basic back-propagation algorithm. They intend to improve the capabilities of the multi-layer network in the aspects like the training time, generalization, etc. Back-propagation has been applied to a wide variety of research applications like conversion of text to speech (NetTalk) [Sejnowski and Rosenberg 1986], optical character recognition system, machine recognition of handwritten characters [Burr 1987], image compression [Cottrell et. al. 1987] and many more. This is supposed to be the most successful network of all existing ones.

2.5.1.4 Grossberg Network

Grossberg Layer:

For Grossberg layer $N_g = N_k W_g$

Where W_g is Grossberg Weight Matrix
N_g is Grossberg Layer Output Vector

So action of each neuron in the Grossberg layer is to, for example, output the value of the weight that connects it to the single Kohonen neuron (Works like an encoder).

Training the Grossberg Layer: is relatively simple.

(i) Calculate Grossberg Outputs (NG)

(ii) Next, each weight is adjusted for example, only if it connects to the Kohonen neuron having a non-zero output. The amount of the weight adjustment is proportional to the difference between the weight and the desired output of the Grossberg neuron to which it connects.

2.5.2 Supervised Networks: Recurrent

2.5.2.1 Hopfield Nets

In 1982, John Hopfield published a most influential paper which drew attention to the associative properties of a class of neural nets. It contained a fundamental statement of a method for analyzing such scheme. The analysis is based on the definition of "energy" in the net and a proof that the net operates by minimizing this energy when settling into stable patterns of operations. He drew attention to two properties of interconnected cells of simple non-linear device (autoassociative).

(i) that the system has stable states which will always be entered if the net is started in similar states (could be noisy input).

(ii) that such states can be created by changing the strength of the interconnection between the cells.

This is nothing but associative memories in Computer jargon. But advantage is that we can have noisy key to index into the complete content. This could lead to novel integrated circuits and novel computing device.

Assumptions: In Hopfields's theory a "neuron i" has two states like MCP neuron. Neuron i receives an input from neuron j with a strength W_{ji}. If $W_{ji} = 0$, it means that i is disconnected from j. Strength W_{ij} is similar to weight W_{ji} in MCP model.

Important assumption is $W_{ij} = W_{ji}$ and $W_{ii} = 0$.

Hopfield nets are normally used with binary inputs. These nets are most appropriate when exact binary representations are possible as with blank and white images where input elements are pixel values, or with ASCII representation of each character. These nets are less appropriate when input values are actually continuous, because a fundamental representation problem must be addressed to convert the analog quantities to binary values.

Model: This net can be used as an associate memory or to solve optimization problems. One version of the net [Hopfield 1982], which can be used as a content addressable memory is shown in figure 10. This net has N nodes containing hard

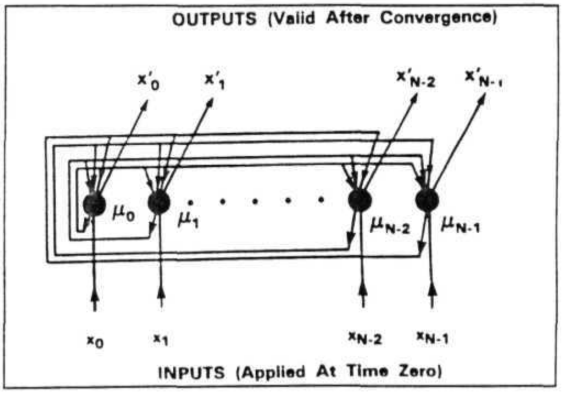


Figure 10. A Hopfield neural net that can be used as a content-addressable memory. An unknown binary input pattern is applied at time zero and the net then iterates until convergence when node outputs remain unchanged. The output is that pattern produced by node outputs after convergence.

Source: Lippmann 1987

limiting non-linearities and binary inputs and outputs taking on the values +1 and -1. The output of each node is fed back to all other nodes via weights denoted as W_{ij}. This net is operated as follows: First, weights are set using the given recipe from exemplar patterns for all classes. Then an unknown pattern is imposed on the net at tine zero by forcing the output of the net to match the unknown pattern. Following this initialization, the net iterates in discrete time steps using the given formula. The net is considered to have converged when outputs no longer change on successive iterations. The pattern specified by the node outputs after convergence is the net output.

Hopfield [Hopfield 1982] and others [Cohen and Grossberg 1983] have proven that this net converges **when** the weights are symmetric ($W_{ij} = W_{ji}$) and node outputs are updated synchronously. Hopfield [Hopfield 1984] also demonstrated that the net converges when activation function similar to the sigmoid non-linearity are used. When the Hopfield net is used as an associative memory, the net output after convergence is used directly as the complete restored memory. When the Hopfield net is used as a classifier, the output after convergence must be computed to the M exemplars to determine if it matches an exemplar exactly. If it does, the output is that class whose exemplar matched the output pattern. If it does not then a "no match" result occurs.

The Hopfield net has three major limitations when used as a content addressable memory. First, the number of patterns that can be stored and accurately recalled is severely limited If too many patterns are stored, the net may converge to a novel spurious pattern different from all exemplar patterns. Such a spurious pattern will produce a "no natch" output when the net is used as a classifier. Hopfield [Hopfield 1982] showed that this occurs infrequently when exemplar patterns are generated randomly and the number of classes (M) is less then 15 times the number of input elements or nodes in the net (N). The number of classes is thus typically kept well below 15N. For example, a Hopfield net for only 10 classes might require more than 70 nodes and more than roughly 5,000 connection weights. A second limitation of the Hopfield net is that an exemplar pattern will be unstable if it shares many bits in common with another exemplar pattern. Here an exemplar is considered unstable if it is applied at time zero and the net converges to some other exemplar. This problem can be eliminated and performance can be improved by a number of orthogonalization procedures. Third limitation is that of hard learning. There are certain patterns for which the network will not converge, hence cannot be stored.

2.5.2.2 Boltzmann Machine

The problem of false well and hardlearning in Hopfield net can be overcome by Boltzmann Machine.

This net was proposed by Hinton [1984] and his colleagues. They had added noise to the Hopfield model, to dislodge the net from a novel false well. Hardlearning was solved by adding a hidden node.

The key to ensuring that the system can escape from local minima lies in the use of "noise". The application of a degree of uncertainty to the energy of the state.

Ludwig Boltzmann, Austrian physicist, discovered that the random motion of the molecules of a gas had an energy directly related to temperature. This effect occurs not only in a gas but in any electronic circuitry which carried a current. High temperatures cause random movements of electrons that change smooth waveforms into rough ones as shown in **the** Figure 11.

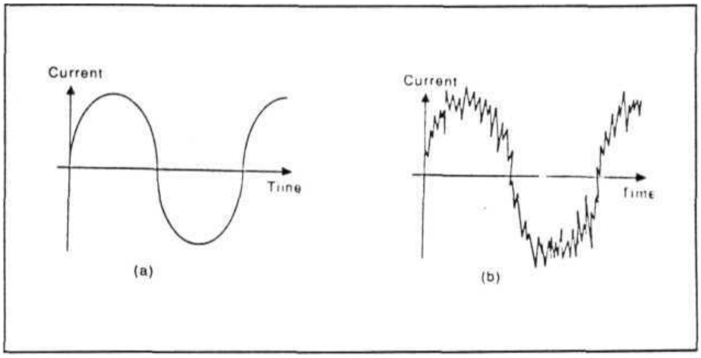


Figure 11. (a) A pure waveform; and (b) A noisy waveform.

Source: Aleksander and Morton 1990

Hinton therefore used the name of Boltzmann to convey the idea that energy of the state of a neural net could be given an uncertainty above and below that which may be calculated as was shown in Hopfield nets. So at Zero temperature, the net is meant to behave exactly like the Hopfield model, while at higher temperature , an uncertainty proportional to the temperature is introduced into the activation function of the net. This may allow it to escape local minima, but it also prevents it from settling anywhere.

Starting at an high temperature and cooling it down while it is running ensures that the state of the net has the best chance of ending in the lowest minima related to given input data. This methodology is called Simulated Annealing. In metallurgy a metal temperature is raised high and cooled slowly to reduce bond stress, and put it in more relaxed state.

2.5.2.3 Bidirectional Associative Memories (BAMS)

In Hopfield nets associativity is strictly speaking, autoassociative, that is, a memory can be completed or corrected, but cannot be associated with different memories. This is a result of their single-layer structure, which requires the output vector to appear on the same neuron on which the input vector was applied.

BAM (Fig. 12) is hetroassociative, i.e., it accepts an input vector on one set of neurons and produces a related, but different, output vector or another set.

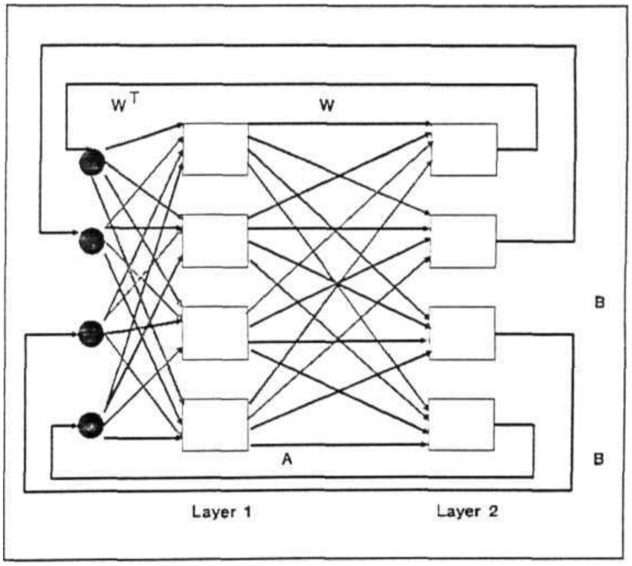


Figure 12. BAM Structure.

BAMs are capable of producing correct outputs despite corrupted inputs.

$$B = F(AW)$$

Where B is the vector of outputs from layer 2
A is the vector of outputs from layer 1
W is the weight matrix between layers 1 and 2
F is the activation function

Similarly $A = F(BW^T)$ W^T is transpose **of** W.

Retrieving **a** stored association: Associations are stored in the weight arrays W and W^T. Each memory consists of two vectors; "A" which appears at the outputs of layer 1 and 2, the associative memory that is the output of layer 2.

To retrieve an associated memory, all or part of vector A is momentarily forced onto the outputs of Layer 1. A is then removed and the network is then allowed to stabilize, producing the associated vector B at the output of layer 2, i.e., the vectors pass back and forth between the layers, always reinforcing the current outputs such that vectors come close to the stored memory till the point that constitutes resonance.

2.5.3 Unsupervised Networks

2.5.3.1 The Carpenter/Grossberg Classifier

This net differs from the Hamming net in that feedback connections are provided to turn off that output node with **a** maximum value, and to compare exemplars to the input for the threshold test required by the leader algorithm. The major components of a Carpeneter/Grossberg classification net with three inputs and two output nodes are presented in Figure 13.

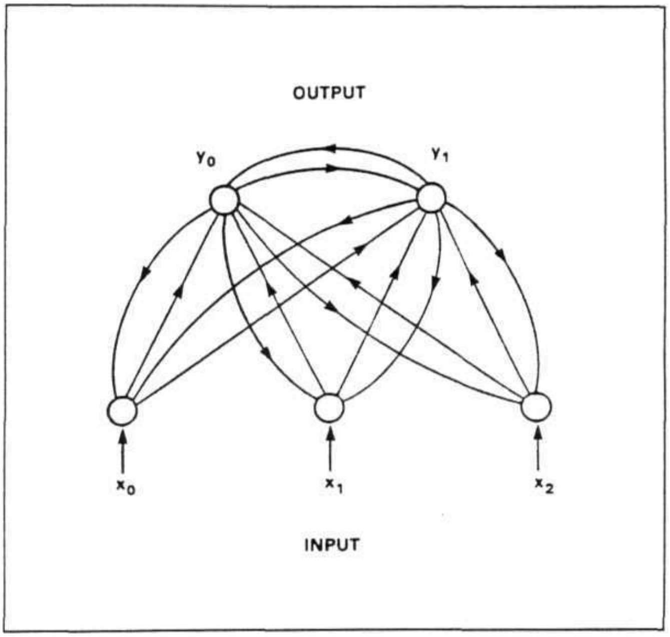


Figure 13. The major components of the Carpenter Grossberg classification net. A binary input is presented at the bottom and when classification is complete only one output is high. Not shown are additional components required to perform the vigilance test and to disable the output node with the largest output.

Source: Lippmann 1987

The algorithm assumes that "fast learning" is used as in the simulations presented by Carpenter and Grossberg [1983] and thus that elements of both bottom-up and top-down connections take on only the values 0 or 1. The net is initialized by effectively setting all exemplars represented by connection weights. In addition, a matching threshold called Vigilance which ranges between 0.0 to 1.0 must be set. This threshold determines how close a new input pattern must be to a stored exemplar to be considered similar. A value near one requires a close match and smaller values accept a proper match. New inputs are presented sequentially at the bottom of the net as in the Hamming net. After presentation, the input is compared to all stored exemplars in parallel as in the Hamming net to produce matching scores. The exemplar with the highest matching score is selected using lateral inhibition. It is then compared to the input by computing the ratio of the dot product of the input and the best matching exemplar (number of l bits in common) divided by the number of 1 bits in the input. If this ratio is greater than the vigilance threshold, then the input is considered to be similar to the best matching exemplar and that exemplar is updated by performing a logical AND operation between its bits and those in the input. If the ratio is less than the vigilance threshold, then the input is considered to be different from all exemplars and it is added as a **new** exemplar. Each additional new exemplar requires one node and connections to compute matching scores.

The Carpenter/Grossberg algorithm can perform well with perfect input patterns but even a small amount of noise can cause problems. However, this level may be too high and the number of stored exemplars can rapidly grow until all available nodes are used up.

2.5.3.2 Kohonen's Self Organizing Feature Maps

One important organizing principle of sensory path ways in the brain is that the placement of neurons is orderly and often reflects some physical characteristic of the external stimulus being sensed [Kandel and Schwartz 1985]. For example, at each level of the auditory pathway, nerve cells and fibers are arranged anatomically in relation to the frequency which elicits the greatest response in each neuron. This organization in the auditory pathway extends up to the auditory cortex [Moller 1983, Kandel and Schwartz 1985]. Although much of the low-level organization is genetically pre-determined, it is likely that some of the organization at higher levels is created during learning by algorithms which promote self organization. Kohonen [Kohonen et. al. 1984] presents one such algorithm which produces what he calls self organizing feature maps similar to those that occur in the brain.

Kohonen's algorithm creates a vector quantizer by adjusting weights from common input nodes to M output nodes arranged in a two dimensional grid as shown in Figure 14. Output nodes are extensively interconnected with many local

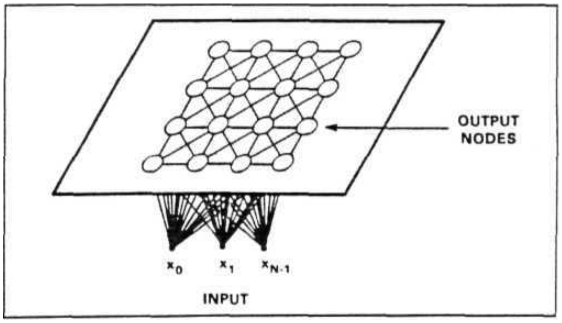


Figure 14. *Two dimensional array of output nodes used to form feature maps. Every input is connected to every output node via. a variable connection weight.*

Source: Lippmann 1987

connections. Continuous-valued input vectors are presented sequentially in time without specifying the desired output. After enough input vectors have been presented, weights will specify cluster or vector centers that sample the input space such that the point density function of the vector centers tends to approximate the probability density function of the input vectors [Kohonen et. al. 1984]. In addition, the weights will be organized such that topologically, close nodes are sensitive to inputs that are physically similar. Output nodes will thus be ordered in a natural manner. This may be important in complex systems with many layers of processing because it can reduce lengths of inter-layer connections.

The algorithm that forms feature maps requires a neighborhood to be defined around each node as shown in Figure 15. This neighborhood slowly decreases in size with time as

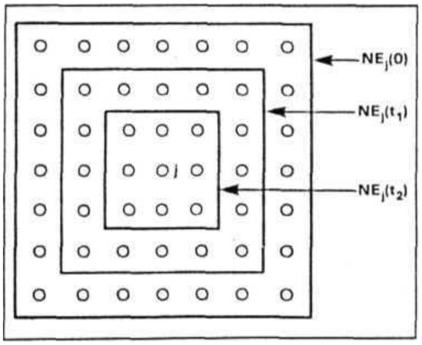


Figure 15. Topological neighborhoods at different times as feature maps are formed, $NE_1(t)$ is the set of nodes considered to be in the neighborhood of node j at time t. The neighborhood starts large and slowly decreases in size over time. In this example, $0 < t_1 < t_2$.

Source: Lippmann 1987

shown. Weights between input and output nodes are initially set to small random values and an input is presented. The distance between the input and all nodes is computed as shown. If the weight vectors are normalized to have constant length (the sum of the square weights from all inputs to each output are identical) then the node with the minimum Euclidean distance can be found to form the dot product of the input and the weights. The selection required turns into a problem of finding the node with a maximum value. Once this node is selected, weights to it and to other nodes in its neighborhood are modified to make these nodes more responsive to the current input. This process is repeated for further inputs, weights eventually converge and are fixed after the gain term is reduced to zero.

Unlike the Carpenter/Grossberg classifier, this algorithm can perform relatively well in noise because the number of classes is fixed, weights adapt slowly and adaptation stops after training. This algorithm is thus a viable sequential vector quantizer when the number of clusters desired can be specified before use and the amount of training data is large relative to the number of clusters desired. It is similar to the K-means clustering on the presentation order of input data for small amounts of training data.

2.5.4 Hybrid Networks

2.5.4.1 Counter Propagation

Counter propogation network (CPN) is inferior to back-propagation for most mapping network applications. However, the network trains rapidly; appropriately applied it can save large amount of computer time.

CPN is combined of two well-known algorithms: the self-organizing map of Kohonen [1988] and Grossberg's outstar **[1982,** 86].

Network structure: This looks much like other Networks **we** have examined so far; however, the difference lies in **the** processing done by the Kohonen and Grossberg neurons.

Normal operation: Kohonen Layer: Function in a "Winner-rake-all" fashion. That is for a given input vector, one and only one Kohonen neuron output a logical one; all others output a zero.

In vector notation is expressed as:

$N_k = XW_k$	W_k	Kohonen Weight Matrix
	N_k	Kohonen Layer Output Vector
	X	Input Vector

Grossberg Layer:

Similarly $N_g = N_k W_g$

Where	W_k	Grossberg Weight Matrix
	N_k	Grossberg Layer Output Vector

So action of each neuron in the Grossberg layer is to output the value of the weight that connects it to the single Kohonen neuron (Works like an encoder).

2.S.4.2 The Hamming Net

The Hopfield net is often tested on problems where inputs are generated by selecting an exemplar and reversing bit values randomly and independently with a given probability [Hopfield 1982, Gold 1986, Wallace 1986]. This is a classic problem in communications theory that occurs when binary fixed-length signals are sent through a memoryless binary symmetric channel. The optimum minimum error classifier in this case calculates the Hamming distance to the exemplar for each class and selects that class with the minimum Hamming distance [Gallager 1968]. The hamming distance is the number of bits in the input which do not match the corresponding exemplar bits. A net which will be called a Hamming net implements this algorithm using neural net components as shown in Figure 16.

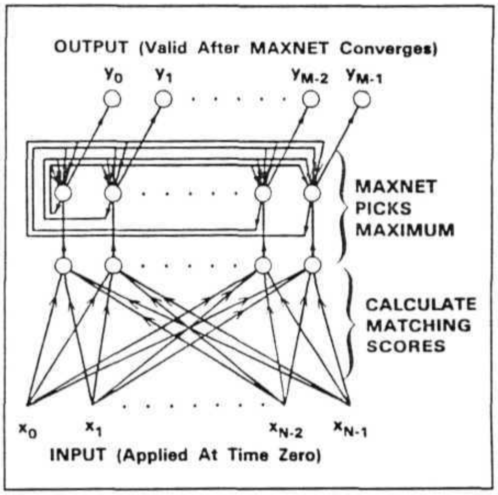

OUTPUT (Valid After MAXNET Converges)
y_0 y_1 y_{M-2} y_{M-1}
MAXNET PICKS MAXIMUM
CALCULATE MATCHING SCORES
x_0 x_1 x_{N-2} x_{N-1}
INPUT (Applied At Time Zero)

Figure 16. A feed-forward Hamming net maximum likelihood classifier for binary inputs corrupted by noise. The lower subnet calculates N minus the Hamming distance to M exemplar patterns. The upper net selects that node with the maximum output. All nodes use threshold-logic non-linearities where it is assumed that the outputs of these non-linearities never saturate.

Source: Lippmann 1987

Weights and thresholds are first set in the lower subnet such that the matching scores generated by the outputs of the middle nodes of Figure 16 are equal to N minus the Hamming distances to the exemplar patterns. These matching scores will range from 0 to the number of elements in the input (N) and are highest for those nodes corresponding to classes with exemplars that best match the input. Thresholds and weights in the MAXNET subnet are fixed. All thresholds are set to zero and weights from each node to itself are 1. Weights between nodes are inhibitory with a value of -e where e < 1/M, where M are the number of nodes in MAXNET.

After **weights** and thresholds have been set, a binary pattern **with** N elements is presented at the bottom of the Hamming net. It must be presented long enough to allow the matching score outputs of the lower subnet to settle and initialize **the** output values of the MAXNET. The input is then removed and the MAXNET iterates until the output of only one node is positive. Classification is then complete and the selected class is that corresponding to the node with a positive output.

The Hamming net has a number of advantages over the Hopfield net. It implements the optimum minimum error classifier **when** bit errors are random and independent and thus the performance of the Hopfield net must either be worse than or equivalent to that of the Hamming net in such situations. Comparisons between the two nets on problems such as character recognition, recognition of random patterns and bibliographic retrieval have demonstrated this difference in performance [Lippmann et. al. in press]. The Hamming net also requires many fewer connections than the Hopfield net. For example, with 100 inputs and 10 classes the Hamming net requires only 1,100 connections while the Hopfield net requires almost 10,000. Furthermore, the difference in number of connections required increases as the number of inputs increases, because the number of connections in the Hopfield net grows as the square of the number of inputs while the number of connections in the Hamming net grows linearly. The Hamming net can also be modified to be a minimum error classifier when errors are generated by reversing input elements from +1 to -1 and from -1 to +1 asymmetrically with different probabilities [Lippmann et. al. in press] and when the values of specific inputs elements are unknown [Baum et. al. 1986]. Finally, the Hamming net does not suffer from spurious output patterns which can produce a "no-match" result.

2.5.5 KBANN: Knowledge-based Artificial Neural Network

KBANN is a network formed by mapping rules into the network. Hence the knowledge (rules in case of knowledge based systems) determines the structures of the artificial neural networks and the weights on its links, make the learning accessible for modification by neural learning [Shavlik 1989].

KBANN uses a knowledge base of hierarchically structured rules v/hich nay be both incomplete and incorrect to form an artificial neural network. Hence the correspondence between the knowledge base and the artificial neural networks can be described as follows.

Knowledge Base	ANN
Final conclusions	Output units
Supporting facts	Input units
Information conclusions	Hidden units
Dependencies	Weighted connections

Hence the given rules of a knowledge base can be mapped into an artificial neural network (Figure 17) . Rules are assumed to be conjunctive, non-recursive and variable free. Disjuncts are coded as multiple rules. Translating rules into artificial neural network. Weights are set on links and biases of units in such a way that units have a significant activation only when corresponding deduction could be made using the knowledge base. Assuming there exists a rule in the knowledge base with n mandatory antecedent(ie, antecedents which must be true) and prohibitory antecedents (i.e. antecedents which must not be true), the system sets Weights W and - W on links in the artificial neural network corresponding to the mandatory and prohibitory dependencies of

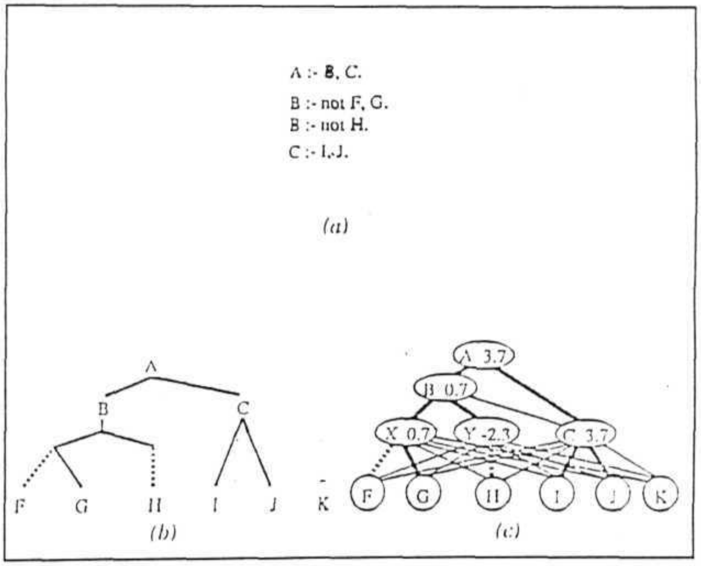


Figure 17. Translation of KB to ANN.

the rule respectively. The bias on the unit corresponding to the rule's consequent is set to n*W - ϕ. ϕ is a parameter chosen so that units have activations 0.9 (approx) when their antecedents are satisfied, and activation 0.1 (approx) otherwise.

Disjunctive rules cannot be transformed directly because there is no way the bias of unit can be set allowing it to get activated in multiple ways such that no unintended combinations are allowed. Hence disjuncts are handled by creating units Y^1 and Y^2, which correspond to rules R^1 and R^2 (R^1 and R^2 being conjunctive rules). These units will be active when their corresponding rule is true. Then R^1 and R^2 are connected to Y by a link of weight W and sets the bias of Y to W - $. Hence Y will be active when either R^1 or R^2 active.

2.5.5.1 Learning in KBANN

KBANN combines empirical and explanation based learning to overcome the problem of each approach by using training examples to inductively refine pre-existing knowledge. Once the initial translation from knowledge base to artificial neural networks is completed, input from knowledge base to artificial neural network is completed, input units corresponding to features of the environment that do not appear as an antecedent are added to the network. These input units night be necessary to express some concept accurately. Links are added to network to give existing rules access to items not mentioned in the knowledge base. These links initially have weight equal to zero. The network is perpetuated by adding random numbers to all link weights and bias to avoid symmetry breaking problems [Rumelhart et al. 1986]. The network is now refined by providing training examples which are processed using back-propagation algorithm [Rumelhart et al. 1986]

2.6 Applications of Neural Computers

2.6.1 Pattern Recognition

The most important application areas for "neural pattern recognition" could be the same as those for which conventional, heuristic methods have been developed during the past thirty years (a) remote sensing, (b) medical image analysis, (c) industrial computer vision (especially for robotics), and (d) input devices for computers.

Most concrete tasks for which special computer equipment has already been developed are: (a) segmentation and classification of regions from images, (b) recognition of handwritten characters and text, (c) recognition of speech, and (d) processing, especially restoration of noisy pictures.

One more ambitious level, one may wish to achieve the capabilities of is: (a) image analysis (referring to different thematic levels of abstraction, such as monitoring of land use on the basis of satellite pictures), (b) image understanding (interpretation of scenes) and (c) speech understanding (parsing and interpretation of spoken sentences).

To implement these tasks, certain basic problems still call for better understanding, for instance, those concerning the intrinsic properties (features) of input information, such as: (a) the most natural pattern primitives(lines, their curvatures and end points, edges, statistics of point groups), (b) visual information which describes the surface curvature and cusps, (c) texture, and (d) phonological invariants in speech.

2.6.2 Knowledge Database for Stochastic Information

To find a solution to a searching task which is defined in terms of several simultaneous incomplete queries, as the case in formal problem solving tasks usually is, it is thus not sufficient to implement a content addressable (or autoassociative) memory, but the partial searching results must somehow be buffered and studied sequentially. This, of course, is completely expedient for a digital computer, which can store the candidates as lists and study them by a program code, in a neural network, however, task like holding a number of candidates in temporary storage, and investigation of their "markings" would be very cumbersome.

In artificial neural network, the searching arguments are usually imposed as initial conditions to the network, and solution for the "answers" results when the activity state of the networks relaxes to some kind of energetic minimum. One has to note the following facts that are characteristics of these devices: (a) Their network elements are analog devices, whereby representations of numerical variables, and their matching can only be defined with relatively low accuracy. This, however may be sufficient for prescreening purpose which is most time consuming: (b) a vast number of relations in memory which only approximately match with the search argument can be activated. On the other hand, since the conflicts then cannot be totally resolved but only minimized, the state of neural network to which it converges in the process represents some kind of optimal answer(usually, however, only in the sense of Euclidean metric): and (c) The "answer" or the asymptotic state which represents the searching result has no alternatives. Accordingly, it is not possible except in some rather weird constructs, to find the complete set of solutions, or even a number of the best candidates for them. It is not sure that the system will converge to the global optimum: it is more usual that the answer corresponds to one of the local optima which, may be an acceptable solution in practice.

2.6.3 Optimization Computations

In general, the objective is to allocate a limited amount of resources to a set of certain partial -asks such as objective or cost function is minimized (or maximized) . A great number of variables usually enter the problem, and to evaluate and to minimize the objective function, a combinatorial problem has to be solved. In large-scale problems such as optimization of economic or business operations, the systems of equations are usually static, although non-linear, and if conventional computers are used, the solutions must be found in a great « many iterative steps.

Another category of complex optimization tasks is met in systems and control problems which deal with physical variables and space and time-continuous processes. Their interrelations(the restricting conditions) are usually expressed as systems of partial differential equations, whereas the objective function is usually an integral-type functional. Mathematically these problems often call for methods of variational calculus. Although the number of variables then may be orders of magnitude smaller than in the first category of problems, exact mathematical treatment of the functionals again creates the need of rather large computing power.

It may come as a surprise that "massively parallel" computers for both of the above categories of problems existed in 1950s. The differential analyzer based on either analog or digital computing principles and components, were set up as direct analogies of the systems to be studied, whereby plenty of interconnections(feedbacks) were involved. For details of these systems and many of the problems already solved by them, see Korn and Korn [1964], Aoki [1967] and Tsypkin [1968].

It may then also be obvious that if the "massively parallel computers" such as "neural networks" are intended to solve optimization problems, they must at least in principle operate as analog devices;the dynamics of their processing elements must be definable with sufficient accuracy and individually for each element, and the interconnectiveness must be specifically configurable.

2.6.4 Robotic Control

There are two main categories of robots; trajectory programmed ones, and so-called intelligent robots. To program the former, a human programmer first controls their movements and actions in the desired way, whereby the sequence of coordinates and commands are stored in memory. During subsequent use, identical trajectories and commands are defined by the memorized information. The intelligent robots are supposed to plan their own actions; typical applications for them are assembly tasks whereby the components have to be located and fetched from random places; or the robots may be freely moving in the natural environment, at the same time performing non-programmed tasks.

The "intelligence" exhibited by the robots has so far been implemented by AI programs, which means that the strategies have to be invented and programmed heuristically by a human being. It is often desirable to have a higher degree of learning in such robots, which then calls for "neural computers". For instance, learning of locomotion in an unknown environment is a task which hardly can be formalized by logic programming, and coordination of complex sensory functions with the motor ones cannot be solved in analytical form. Some computer simulations have already been performed which have demonstrated such autonomous learning capabilities [e.g. Barto et.al. 1983].

2.6.5 Decision Making

A more abstract and complex version of behavior which nonetheless belongs to the same category as the robot operations is the non-rule-based decision making, eventually connected with playing games. In the conventional AI implementations, the conditions and actions entering the problem are described as decision tree, the evaluation of which is a combinatorial problem, and for the solution of **which** the branches have to be studied up to a certain depth. This, however, is exactly not a way in which a natural object thinks. He may make formal analyses in order to avoid bad decision, but Then it comes to the final strategy, then other reasons, based on hunches and intuitive insight into the situation become more important. These capabilities, however may result in sufficiently large artificial learning systems which operate according to "neural computing principles". The performance criterion thereby applied is more complex, although implicit, and it will be learned automatically from examples as some kind of higher order statistical description. It may then be learned automatically from examples as some kind of high-order statistical description. It may then be said that such strategies are also stored in the form of rules, where as these rules are established automatically, and they only exist in implicit form, as the collective states of the adaptive interconnections.

Chapter 3

Expert System and Knowledge-based Artificial Neural Network

Expert Systems such as Mycin, Dendral, Prospector, Caduceus, etc., proved to be successful in early eighties. In late eighties success of the Neural Network (NN) approach to problems such as learning to speak [Sejnowski and Rosenberg 1986], medical reasoning [Gallant 1988], recognizing handwritten characters, [Mui and Agarwal 1993; Baxt 1992; Wang 1991; Bottan and Vapnik 1992; Hoffman et.al 1993; Nagendra Prasad et.al 1993] etc., gave a new impetus to research in Neural Computing. The neural network approach has been an increasingly important approach to artificial intelligence [Feldman and 3allard 1982; Rumelhart et al. 1986; Smolensky 1987; Feldman et al. 1988]. The Neural Network approach is being applied to difficult, A.I. problems. Fu and Fu [1990] suggested a novel approach wherein a rule-based conventional expert system was mapped into a neural architecture in both the structural and behavioral aspects. It was suggested that the NN approach can enhance the performance of Conventional Expert Systems (CES).

The Neural Network approach contrasts with the knowledge-based approach in several aspects. The knowledge of a neural network lies in its connections and associated weights, whereas the knowledge of a rule-based system lies in rules. A neural network processes information by propaga-ting and combining activations through the network, but a knowledge-based system (Fig. 1) reasons through symbol generation and pattern matching. The knowledge-based approach emphasizes knowledge representation, reasoning strategies and the ability to explain, whereas the neural network approach does not. The key differences between these two approaches are summarized in Table 1.

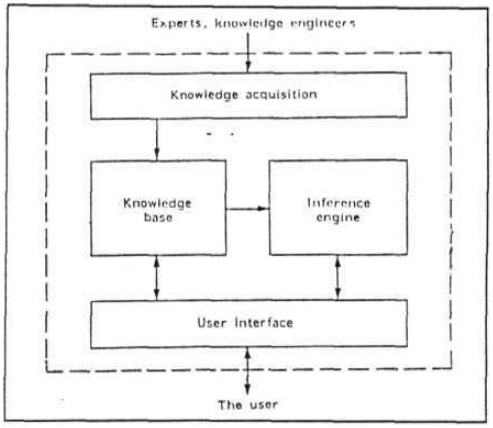


Figure 1. The basic components of a knowledge-based system.

Source: Fu and Fu 1990

Table 1. Comparison between the neural network and the knowledge-based approaches.

	Neural network approach	Knowledge-based approach
Knowledge	Connections	Rules
Computation	Numbers Summation and thresholding Simple, uniform	Numbers, symbols, pattern matching Complicated, various
Reasoning	Non-strategic	Strategic meta-level
Tasks	Signal level	Knowledge level

3.1 Mapping Rule-based Systems into Neural Architecture

A rule-based system (knowledge represented in rules) can be transformed into an inference network where each connection corresponds to a rule and each node corresponds to the premise or the conclusion of a rule, as seen in Figure 2. Reasoning

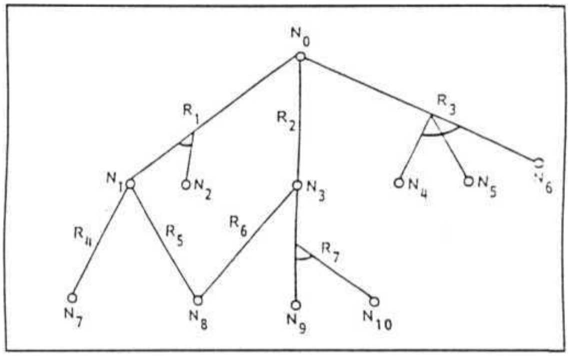


Figure 2. An inference network.

Source: Fu and Fu 1990

in such systems is a process of propagating and combining multiple pieces of evidence through the inference network until final conclusions are reached. Uncertainty is often handled by adopting the certainty factor (CF) or the probabilistic schemes which associate each fact with a number called the belief value. An important part of reasoning tasks is to determine the belief values of the pre-defined final hypothesis given the belief values of observed evidence. The network of an inference system through which belief values of evidences or hypotheses are propagated and combined is called the belief network. Correspondence in structural and behavioral aspects exists between neural networks and belief networks, as shown in Table 2. **For** instance, the summation function in neural networks corresponds to the function for the Bayesian formula for deriving posterior probabilities in PROSPECTOR-like systems. The thresholding function in neural networks corresponds to predicates such as SAME (in Mycin-like systems), which cuts off any certainty value below 0.2.

Table 2. Correspondence between neural networks and belief networks.

Neural networks	Belief networks
Connections	Rules
Nodes	Premises, conclusions
Weights	Rule strengths
Thresholds	Predicates
Summation	Combination of belief values
Propagation of activations	Propagation of belief values

Since belief network corresponds to neural network by mapping the knowledge base and the inference engine into a kind of neural network called conceptualization, which stores knowledge and performs inference and learning. Furthermore, to construct a conceptualization, the following mappings need to be done.

- Final hypotheses are mapped into output neurons (neurons without connections pointing outwards),

- Data attributes are mapped into input neurons (neurons without connections pointing outwards),

- Concepts that summarize or categorize subsets of data or intermediate hypotheses that infer final hypotheses are mapped into middle (also known as hidden) neurons, and

- The strength of a rule is mapped into the weight of the corresponding connection.

If there are no data errors, input neurons can represent both the observed and the actual data. In case of possible data errors, the observed data and the actual data are represented by two different levels of neurons, with a connection established between each observed and actual input neurons referring to the same data attribute. One example is shown in Figure 3, where for instance, observed input neuron E_1 corresponds to actual input neuron E_1.

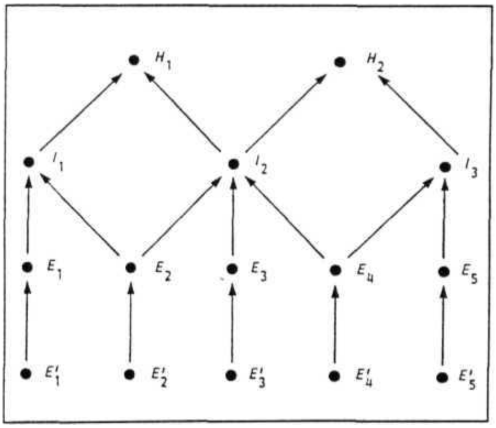


Figure 3. *Organization of the knowledge-base* ***and*** *input data as a neural network.*

3.2 Knowledge Representation

In this section the knowledge representation language in MYCIN [Buchanan and Shortliffe 1934] or similar systems is reviewed. The issue of how to map such language into conceptualization is then examined, and knowledge representation of the neural network is described.

In MYCIN, facts are represented by context-attribution (or object-attribute-value) triples. Each triple is a term. For instance, the term 'throat which is the site of the culture' is represented by the triple <CULTURE SITE THROAT>. Each triple is associated with certainty factor, which is described later.

A sentence is represented by predicate-context-attribute-value quadruple. For instance, the sentence 'the site of the culture is throat' is represented by quadruple <SAME CULTURE SITE THROAT>. The truth value of a sentence is determined by whether the triple satisfies the predicated in terms of its CF.

Judgmental and inferential knowledge is represented in production rules; i.e., if-then rules. If a rule's IF-part is evaluated to be true, its THEN part will be concluded. Each part is constituted by a small nur.ber of sentences. For instance, a MYCIN rule.

*RULE 12 4
IF:
1. The site of the culture is throat.
2. The identity of the organism is Streptococcus.
THEN: There is strongly suggestive evidence (.8)
that the subtype of the organism is not group-D.

can be encoded in MYCIN language as

```
*(RULE 124 (($AND(SAME CULTURE SITE THROAT)

 (SAME ORGANISM IDENTITY STREPTOCOCCUS))

 ((CONCLUDE ORGANISM SUBTYPE GROUP-D -.8))))
```

Certainty factors are integers ranging from -1.0 to 1.0. A minus number indicates disbelief whereas a positive number indicates belief. The degree of belief or disbelief parallels the absolute value of the number. The extreme values -1.0 & 1.0 represent 'No' and 'Yes' respectively. A triple is associated with a CF indicating the current belief in the triple. A rule is assigned a CF representing the degree of disbelief in the conclusion given the premise is true. For instance, the CF of RULE in the example above is -0.8. The CF of a conclusion based upon rule can be computed by multiplying the CF of the premise and the CF of the rule. Each sentence or condition in the premise on evaluation will return a number ranging from 0 to 1.0 representing the CF of the sentence. The CFs of all conditions in the premise are combined to result in the CF of the premise. As in the fuzzy set theory, $AND returns the minimum of the CFs of its arguments. CFs of a fact due to different pieces of evidence are combined according to certain formulae.

A sentence in the rule language is mapped into a concept node (a node in the conceptualization) . Mapping at this level of abstraction can capture the analogies between a belief and a neural network shown in Table 2. Mappings at lower levels, such as mapping a word in a sentence into a concept node lack a good justification.

Suppose the premise of a rule involves conjunction, then each sentence in the premise is mapped into a concept node. These concept nodes then lead into another concept node representing the conjunction.

The CF of a sentence is mapped into the activation level of the concept node designated by the sentence. The CF of a rule is mapped onto the weight of the connection between the two concept nodes, one designated by the premise and the other by the conclusion of the rule.

A neural network is a directed graph where each arc is labeled with a weight. Therefore, it is defined by a two-tuple (V,A) , where V is a set of vertices and A is a set of arcs. The knowledge of a neural network is stored in its connections and weights. The data structure to represent a neural network should take into account how to use its knowledge. Here the scheme used to represent a neural network will be described.

Assume that the network is arranged as multiple layers. Each layer contains a certain number of nodes (processing elements). A node receives input from some other nodes which feed into the node. If node A leads into node B, we say that node A is adjacent to node B and node B is adjacent to node A. There is one list from each node in the network. The members in list i represent the nodes that are adjacent to node i. To make the access to these lists fast, all the nodes are stored in an array where each node points to the list associated with it, as shown in Figure 4. This scheme is known as 'inverse adjacency lists' in graph theory. Connection weights are stored in properly defined data fields in the adjacency lists. Since the activation level at a given node is computed based on the activations at the nodes adjacent to the node, inverse adjacency lists offer

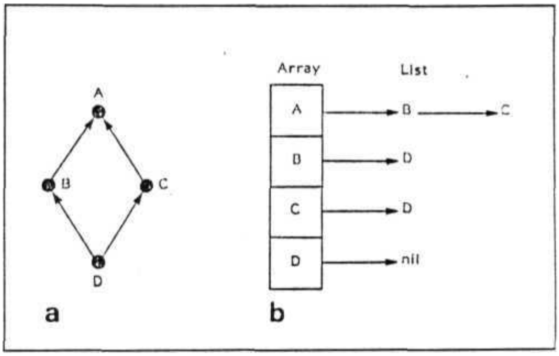


Figure 4. Representation of a neural network (a) as inverse adjacency lists (b).

Source: Fu and Fu 1990

computational advantages. By contrast, the scheme of adjacency lists which contain nodes adjacent from a given node is useful for back-propagation.

3.3 Inference

Inference in most rule-based systems is to deduce the CFs of pre-defined hypotheses from given data. Such systems have been applied successfully to several types of problems such as diagnosis, analysis, interpretation and prediction. MYCIN uses a goal-oriented strategy to make inference. This means it invokes rules whose consequents deal with the given goal and recursively turns a goal into subgoals suggested by the antecedents of rules. By contrast, a system which adopts a data-driven strategy will select rules whose antecedents are matched by the database. Despira the difference between the rule selection between these two strategies, inference in rule-based system is a process cf propagating and combining CFs through the belief network. Since inference in the neural network involves a similar process, with CFs replaced by activation levels, the formulae for computing CFs can be applied to compute the activaticr. level at each concept node in the conceptualization.

If a rule-based system involves circularity (cyclic reasoning), then inference in the neural networks mapped by such a system is characterized by not only propagation and combination of activations but also iterative search for a stable state, if it converges, in an extremely short period of time measured at the unit *si* the time constant of the neural circuit.

The inference capability of the neural network is derived from the collective behavicr of simple computational mechanisms at individual nodes. The output of a node is a function of the weighted sum of its inputs. In a biological neuron, if and only if its input exceeds a certain threshold, the neuron will fire. For an artificial neuron, continuous non-linear transfer functions such as the sigmoid function and non-continuous ones such as threshold logic have been defined. A neural network is often arranged as single-layered or multi-layers::, and is organized as feedforward or with collateral or recurrent circuits. Different architectures are taken in accordance with the problem characteristics.

In a feedforward neural network as discussed in the previous chapter the inference behavior is characterized by propagating and combining activations successively in the forward direction from input to output layers. Collateral inhibition and feedback mechanisms are implemented using collateral and recurrent circuits, respectively. They are employed for various purposes. For instance, the winner-take-all strategy can be implemented with collateral inhibition circuits. Feedback mechanisms are important in adaptation to the environment. As to the layered arrangement, multi-layered neural networks are more advantageous than single-layered networks in performing non-linear classification. This advantage ster.s from the non-linear operation at the hidden nodes. For instance, exclusive-OR can be simulated by a bi-layered neural network but not by any single-layered one. The principle of maximum information preservation (informax principle) has been proposed for information transformation from one layer to another in a neural network [Linsker 1988]. This principle can shed light on the design of a neural network for information processing.

The inference tasks performed by the neural network generally fall into four categories: pattern recognition, association, optimization and self-organization. A single-layered network can act as a linear discriminant, whereas a multi-layered network can be an arbitrary non-linear discriminant. Association performed by the neural network is content-directed allowing incomplete matching. Optimization problems can be solved by implementing cost function as neural circuits and optimizing them. Self-organization is the way the neural network evolves unsupervisedly in response to environmental changes. Clustering algorithms can be implemented by neural networks with self-organization abilities. (See previous chapter for details).

MYCIN like expert systems will be mapped into neural networks which are in general feedforward and multi- layered, and perform tasks close to pattern recognition. By capitalizing on all inference capabilities of neural networks, it is possible to develop expert systems more versatile than the existing ones.

3.4 Learning

Learning in the conceptualization is the process of modifying connection weights to achieve correct inference behavior. The following will show how to apply the back-propagation rule to learn and how to revise rules and/or data on the basis of the results through learning.

In a knowledge-based system, the issue of learning deals with acquiring new knowledge and maintaining integrity of the knowledge base. The knowledge base is constructed through a process called knowledge engineering (encoding of expert knowledge) or through machine learning.

When errors are observed in the conclusions made by a rule-based system, an issue is raised of how to identify and correct the rules or data responsible for these errors. The problem of identifying the sources of errors is known as the blame assignment problem.

Previous approaches [Poitakis 1982; Suwa et al. 1984; Wilkins and Buchanan 1986] only focus on how to revise a rule-based system. Among these TEIRESIAS [Davis 1976] is a typical work. It maintains the integrity of knowledge base by interacting with experts. However, as the size of the knowledge base grows, it is no longer feasible for human experts to consider all possible interactions among knowledge in a coherent and consistent way. TMS [Doyle 1979] resolves inconsistency by altering a minimal set of beliefs, but it lacks the notion of uncertainty in the method itself. Symbolic machine learning techniques such as the RL program [Fu 1985] can learn and debug knowledge but in general do not address the case when the knowledge involves intermediate concepts which are not used to describe the training samples.

TEIRESIAS may be confronted with the following problems. First, incorrect conclusions may be due to data errors. Second experts know the strengths of inference for each individual rule, but it may be difficult for them to determine the rule strengths in such a way that dependencies among rules are carefully considered to meet the system assumptions. For instance, in MYCIN, since certainty factors are combined under the assumption of independence, the certainty factors assigned to two dependent rules should be properly adjusted so as to meet this assumption.

3.4.1 Back Propagation of Error

An error refers to the disagreement between the belief values generated by the system and that indicated by a knowledge source assumed to be correct (eg., an expert) with respect to some fact. The back propagation rule developed in the neural network approach [Rumelhart et.al 1986] is a recursive heuristic which propagates backwards errors at a rule to all nodes pointing to that node, and modifies the weights of connections heading into nodes with errors. First we will restrict our attention to single-layered networks involving only input and output neurons.

In each inference task, the system arrives at the belief values of final hypotheses given those of input data. The belief values of input data form an input pattern (or an input vector) and those of final hypotheses form an output pattern (or an output vector) system error refers to **the** case when incorrect output patterns are generated by the system. When the system error arises, we use the instance consisting of the input pattern given for inference and the correct output pattern to train the network. The instance is repeatedly used to train the network until a satisfactory performance is reached. Since the network may be incorrectly trained by that instance, we also maintain a set of reference instances to monitor the learning process. This reference set is consistent with the knowledge base. If, during learning, some instances in the reference instances set become inconsistent, they will be added to the learning process.

On a given trial, the network generates an output vector given the vector of the training instance. The discrepancy obtained by. subtracting the network's vector from the desired output vector serves as the basis for adjusting the strengths of the connections involved. The back-propagation rule adapted from [Rumelhart et.al (1986) is formulated as follows.

$$\Delta W_{ji} = rD_j(dO_j / dW_{ji}) \tag{1}$$

where $D_j = T_j - O_j$, ΔW_{ji} is the weight (strength) adjustment of the connection from input node i to the output node j, r is a trial independent learning rate, D_j, is the discrepancy between the desired belief value (T_j) and the network's belief value (O_j) at node j, and the term dO_j / dW_{ji} is the derivative of O_j with respect to W_{ji}. According to this rule the magnitude of weight adjustment is proportional to the product of the discrepancy and the derivative above.

The back-propagation rule is applicable to belief networks where the propagation and the combination of belief values are determined by differentiable mathematical functions. As shown in equation (1), the mathematical requirement for applying the back-propagation rule is that the relation between the output activation (O_j) and the input weight (W_{ji}) is determined by a differentiable function. In belief networks, this relation is differentiable if the propagation and the combination functions are differentiable. Since combining belief values in most rule-based systems involves such logic operations as conjunction or disjunction, the back-propa-gation rule is applied after turning the conjunction operator into multiplication and the disjunction operator into summation.

A multi-layered network [Jones and Hoskins 1987] involves at least three levels; one level of input nodes one level of output nodes and one or more levels of middle nodes. Learning in a multi-layered network is more difficult because the behavior of the middle nodes is not directly observable. Modifying the strengths of the connections pointing to a middle node entails the knowledge of the discrepancy between the network's value and the desired belief value at the middle node. The discrepancy at a middle node can be derived from the discrepancies at output nodes which receive activations from the middle node. It can be shown that the discrepancy at middle node j is defined by

$$D_j = \sum_{k} (dO_k/dO_j) D_k$$

where D_k is the discrepancy at node k, In the summation, each discrepancy D_k is weighted by the strength of the connection pointing from middle node j to node k. This is a recursive definition in which the discrepancy at a middle node is always derived from discrepancies at nodes at the next higher level.

3.4.2 Distinguishing Knowledge-base from Input Data Errors

A method has been devised that can distinguish knowledge base errors fron input data errors. This method includes three tests. In the first test, we clamp all connections corresponding to the knowledge base so that only the strengths of the connection between the observed and the actual input data nodes remain adjustable during learning. In the second test, we clamp the connections between the observed and the actual inputs and allow only the strengths of the connections corresponding to the knowledge base to be modified. In the third test, we allow the strengths of all connections to be adjusted. In each test, success is reported if the error concerned can be resolved after learning; failure is reported otherwise. Consequently there are eight possible outcomes as shown in Table 3. Outcome 01 suggests the revision of either the knowledge base or output

Table 3.

Test1	Test2	Test3	Outcome
S	S	S	01
S	S	F	02
S	F	S	03
S	F	F	04
F	S	S	05
F	S	F	06
F	F	S	07
F	F	F	08

S = success, F = failure

data. In this case, an expert's opinion is needed to decide, which should be revised. Outcome 02 is ignored. Outcone 03 suggests the revision of output data. Outcome 04 is unlikely and is ignored. Outcome 05 suggests the revision of the knowledge base. Outcome 06 is also unlikely and is ignored. Outcome 07 suggests the revision of both the knowledge base and data. Outcome 08 is a deadlock which demands an expert to resolve.

3.4.3 Revision Operations

The revision of the above tests will indicate whether the knowledge base or input data(or both) should be revised. The strengths of the connections in the network (representing the knowledge base and input data) have been revised after learning. The next question is how to revise the knowledge base and/or input data according to the revisions made in the network. The revision of the knowledge base will be dealt with first.

Basically, there are five operators for rule revision

- modification of strengths
- deletion
- generalization,
- specialization, and
- creation

However not all the five operators are suitable in the neural network approach to editing rules. Each operator is examined below.

The modification of operator strengths is straight-forward since the strength of a rule is just a copy of the weight of the corresponding connection and the weights of connections have been modified after learning with the back-propagation rule. If the weight change is trivial, we just keep the rule strength before learning.

The deletion operator is justified by Theorem 1.

Theorem 1: In a rule-based system if the following conditions are met:

1. the belief value of the conclusion is determined by the product of the belief value of the premise ar.d the rule strength.

2. the absolute value of any belief value and the rule strength is not greater than 1, and

3. any belief value is rounded off to zero if its absolute value is below threshold k (k is a real number between 0 and 1).

then the deletion of rules **with** strengths below k will not affect the belief values of the conclusions arrived at by the system.

Proof: Frcr. condition 1 and 2 if the strength of rule R is below k, the belief value of its conclusions is always below k. From condition 3, the belief value of the conclusion made by rule R **will** always be rounded off to zero. Since rule R is not effective in making any conclusion, it can be deleted. Thus the deletion of such rules as rule R will not affect the system conclusions.

Accordingly deletion of rule is indicated when its absolute strength is below the predetermined threshold. In MYCIN-like system, the threshold is 0.2

The deletion operator is also justified by the following argument. Suppose we add some connections to a neural network that has already reached an equilibrium and assign weights to these added connections in such a way that incorrect output vectors are generated. Thus, these conditions are semantically inconsistent. Then, if we train the network with correct samples, the weights of the added connections will be modified in the direction of minimizing their effect. What happens is that the weights will go towards zero and even cross zero during training. In practice, we set a threshold so that when the shift towards zero for a connection weight is greater than this threshold, we delete the connection.

Generalization of a rule can be done by removing some conditions from its premise, whereas specialization can be done by adding more conditions to the premise. If the desired belief value of a conclusion is always higher than that generated by the network and the discrepancy resists decline during learning, it is suggested that rules supporting these conclusions be generalized. Or on the other hand, if the discrepancy is negative and resistant, specialization of a rule may involve qualitative changes of nodes. The back propagation rule has not yet been powerful enough to make this kind of change except deletion of conditions for generalization.

Creation of new rule involves establishment of new connections. Whereas we delete a rule if its absolute strength is below a threshold, we may establish a new connection when its absolute strength is above the threshold. To create new rules we need to create some additional connections which have the potential to become rules. Without any bias, one may need an inference network where all data are fully connected to all intermediate hypotheses, which in turn are fully connected to all final hypotheses. This is not a feasible approach unless the system is small.

From the above analyses, we allow only the modification of strengths and deletion operators in the neural network approach to rule revision.

Revision of input data is much simpler. If the weight of the connection between an observed and an actual input node after learning is below a predeternined threshold, or the shift towards zero is above a certain value, the corresponding input data attribute is treated as false and deleted accordingly.

It has been known that noise associated with training instances will affect the quality of learning. In the neural network approach, since noise will be distributed over the network, its effect on individual connections is relatively minor. In practice, perfect training instances are neither feasible nor necessary. As long as most instances are correct, a satisfactory performance can be achieved.

The comparison between the TEIRESIAS approach and the neural network approach to error handling is shown in table 4. The neural network approach may be more useful than TEIRESIAS in handling multiple errors or errors involving some unobservable concepts which human experts may have difficulties in dealing with. In addition, the back propagation rule can be uniformly applied to the whole rule base, whereas human experts may focus on certain parts of the rule base consciously or subconsciously. Also Wilkins and Buchanan [1986], suggested that the only proper way to cope with deleterious interactions among rules is to delete offending rules. In light of this view the deletion operator **is very** useful. While the neural network approach is still too simple to deal with errors involving qualitative changes of rules, reasoning strategies or meta-level known edge. techniques developed under this approach can supplement the current rule base technology.

Table 4. Comparison between TEIRESIAS and the neural network approach.

	TEIRESIAS	Neural network approach
Approach	Hunan experts	Back-propagation
Operators	Modifying strengths Deletion Addition Generalization Specialization	Modifying strengths Deletion
Errors	Rule errors	Rule and data errors

3.5 Tuning a Rule-base Using Neural Nets

Shavlik & Towell [1989] have given their correspondence for a knowledge base and artificial neural network as shown in tr.a Table 5.

Table 5. Shavlik's correspondence between KB & ANN.

Knowledge base	Neural network
Final conclusions	Output units
Supporting facts	Input units
Intermediate conclusions	Hidden units
Dependencies	Weighted connections

Knowledge based artificial neural network (KBANN) uses a knowledge base (KB) of domain specific inference rules in the form of PROLOG-like clauses to define what is initially known about a topic. The KB need be neither complete nor correct, but needs to only support approximately correct explanations. KBANN translates the KB into an artificial neural network (ANN) in which units and links in the ANN correspond to parts of the KB as shown in the Table 5.

3.5.1 Translation of Rules

Rules are assumed to be conjuctive, non-recursive and variable-free; disjuncts are encoded as multiple rules. The KBANN method sets weights on links and biases of units so that, units have significant activation only when the corresponding deduction could be made using the KB. For example, assume there exists a rule in the KB with n mandatory antecedents (which must be true) and m prohibitory antecedents (which are not true). The system sets weights on links in the ANN corresponding to the mandatory and prohibitory dependencies of the rule to w and -w respectively. The bias on the unit corresponding to the rule's consequent is set to n*w-f, f is chosen such that units have activation approximately 0.9 when their antecedents are satisfied and activation of approximately 0.1 otherwise.

KBANN handles disjuncts by creating units L_1 and L_2 **which** correspond to R_1 and R_2, using the approach for conjunctive rules described above. These units will only be active **when** their corresponding rule is true. KBANN then connects L_1 and L_2 to L by a link of these weight w and sets the bias of L to w -f. Hence, L will be active when either L_1 or L_2 is active.

This concept is explained by means of an example by Towell et.al [1990].

3.5.2 Overview of the KBANN Algorithm is as Follows

1. Translate rules to set initial network structure.

2. Add units not specified by translation.

3. Add links not specified by translation.

4. Perturb the network by adding near zero random numbers to all link weights and biases.

3.5.3 Limitations in Shavlik's Approach

a) They have assumed certainty factors of all premises (including condition & action part) and rule strengths to be 1. i.e., they have proposed logical reasoning using NN.

b) They assumed an output of a neuron to be a binary feature (either 0 or 1) . But in the real case, when we are considering uncertainty factors which are not equal to 1, it will not be so. While they just mentioned about non-binary features but did not elaborate any further.

c) To handle disjunctive rule, a new node has to be inserted in a NN.

Note: Obviously, non-binary features can be used to implement plausible reasoning.

3.6 Inducing Rules for a Connectionist ES

Gallant has implemented a two-program package for constructing connectionist expert systems from training examples. The first program is a network knowledge base generator that uses several connectionist learning techniques, and the second (MACIE) is a stand alone expert system inference engine that interprets such knowledge bases. [Gallant 1988]

3.6.1 Network Properties

A connectionist model consists of a network of (more or less) autonomous processing units called cells that are joined by directed arcs. Each arc ("connection") has a numerical weight $w_{i,j}$ that roughly corresponds to the influence of cell u_j on cell u_i. Positive weights indicate reinforcement; negative weights represent inhibition. The weights determine the behavior of the network, playing some what the same role as a conventional program. They classified networks as either feed forward networks if they do not contain directed cycles or feed-back networks if they do contain such cycles.

Every cell u_i (except for input cells) computes its new activation u, as a function for the weighted sum of the inputs to cell u, from directly connected cells.

$$S_i = \sum^{n} w_{ij}u_j \text{ for } j = 0 \text{ to } j = n$$

$$u, = f(S_i)$$

If u_j is not connected to u_i, then $w_{i,j} = 0$. By convention there is a cell u_0 whose output is always +1 that is connected to every cell u_i (except for input cells). The corresponding weights $(W_{i,0})$ are called biases.

They have given a sample problem for diagnosis and treatment of acute Sachrophagal disease [Buchanan and Shortliffe 1985] .

To generate the connectionist knowledge base, they have used following specifications:

- Name of each cell corresponding to variable of interest (symptoms, diseases, treatments). Each variable will correspond to a cell u_i.
- A question for each input variable, to elicit the value of that variable from the user.
- Dependency information for intermediate variables (diseases) and output variables (treatments). Each of these variables has a list of other variables whose values suffice for computing it.
- The final information supplied to the learning problem is the set of training examples.

They have developed a procedure called pocket algorithm that generates weights for discrete networks. Training algorithm specifies the desired activations for intermediate and output cells in the network(easy learning).

W*: for cell u let $\{E_k\}^{k}$ be the set of training example activations.

$\{C_k\}^{k}$ be the corresponding correct activations for u.

Pocket algorithm is a modified perceptron algorithm. It computes perceptron weight vectors, P, which occasionally replace pocket weight vectors w*.

They have defined rule as an example E_r with the corresponding classification C_r that must be satisfied by the resulting w* Gallant has named his inference engine as MACIE - Matrix Controlled inference engine. It is represented internally by weight matrix.

3.6.2 ES Algorithms

Initial information – the program starts by listing for the user all variables and allowing any input variable to be initialized to true or false.

Inference/forward chaining: It is usually possible to deduce the activation for cell u, without knowing the values of all of its inputs.

Addition of a new rule: Directly contradiction values $E^r = E^s$ but $C^r <> C^s$ are not allowed.

3.6.3 Limitations in Gallant's Approach

a) MACIE is an impossible model for reasoning.

b) Gallant worked with discrete connectionist models.

There are many incomplete areas left in this work.

3.7 Present Work

If is by now quite apparent from the combination of limitations of the earlier three approaches that they have major drawbacks as discussed earlier and the following enhancements would be very effective and useful:

a) Combination and propagation of non-binary belief values in neural networks (rule wise).

b) Evidential reasoning in a network (rule wise).

c) Learning/Training: Process of training the belief values such that net reaches final conclusion **with** desired result.

d) Consistency: Defining the consistency of the rule base in the connectionist expert system.

e) Learning a new rule: The process of learning new rules without any major changes to the previous neural net states, etc.

In the next Chapter we will discuss about how we have made some of the enhancements mentioned above.

Chapter 4

Handling Uncertainty Using ANN for Non-binary Inputs

4.1 Reasoning under Uncertainty

Much of knowledge which humans reason with, is inadequate in some respect or other. Some times a problem **will** necessitate probabilistic assessment of decision. In addition, a knowledge engineer may wish to attach confidence measurements (or lack of confidence) for both hypotheses and conclusions. Problems such as these require that the expert system be capable of dealing with knowledge having varying grades of certainty. Because the world does not behave in a strictly Bayesian or Stochastic fashion, a number of expert systems exploit decision theory to supplement the inference process.

Various theories have been developed to accommodate such uncertainty. To adopt an uncertainty paradigm for NN it is important that we discuss and assess some important theories. We will now discuss four important theories and assess them.

4.1.1 Probability Theory

The mathematical theory of probability provides means of dealing with knowledge about truly random events. This theory includes the following law, among others.

Law: If the probability of A is P(A) **and** the probability of B is P(B) and A and B are independent events then the **probability** of A and B is P(A) * P(B).

Bayes' rules: Bayes' rule provides a way of computing the probability of a hypothesis being true given some evidence related to that hypotheses. The subjective Bayes' approach **was** used in PROSPECTOR.

(a) If P $(E|H_i)$ is the probability that evidence **E will be** observed given that hypothesis H_i is true. For example, E might be symptom and H_i a disease.

(b) P (H_i) is a prior probability that H_i is true. **If H_i** were **a** disease then $P(H_i)$ is the probability of any person having that disease.

(c) K is the number of possible hypotheses which display evidence for E. For example, K night be the number of diseases which display some symptom E. Given these definitions, Bayes' rule states that the probability that hypothesis H_i is true given evidence E denoted by $P(H_i|E)$ may be calculated as

$$P(H_i|E) = (P(E|H_i) * P(H_i)) \;/\; \sum_{n} (P(E|H_n) * P(H_n))$$

Disadvantages of probability theory: There are several disadvantages of using probability theory to deal with uncertainty some of which are mentioned below:

(a) It is often difficult to obtain exact values **for** appropriate probabilities.

(b) It is difficult to modify a Bayesian based set of values because of the dependencies between them.

(c) The single probability value assigned to a hypotheses tells us about its precision.

(d) The single value combines evidence for and against a hypothesis without indicating how much there is of each.

4.1.2 Dempster/Schafer Theory of Evidence

In this approach a distinction is made between uncertainty and ignorance. Instead of probabilities, one specifies belief function, by which one can put bounds on **the** assignment of probabilities to events instead of having to specify the probabilities exactly. The theory also provides methods for computing belief functions for combinations of evidence. When bounds determine the probabilities exactly, this theory reduces to probability theory.

Disadvantages As the Dempster/Schafer theory includes probability theory as a special case, it inherits many of the problems associated with probability theory. It is important since it illustrates the effect of ignorance on reasoning with uncertain knowledge.

4.1.3 Fuzzy Logic

The use of fuzzy logic is gaining popularity due to the following reasons.

(1) It is a formalism that allows for and encourages the use of English language phrases as the means of interaction with the user and is able to deal with ar.d process these English quantifiers in a structured and consistent way; and

(2) That there is no broad assumption of complete independence of the evidence or ideas ro be combined,

such as the one required for the Bayesian and Dempster/Schafer approaches.

The above traits allow a more natural and less contrived tool for encoding knowledge about engineering tasks, while still making a mathematical formalism for combining and reasoning with the inherent uncertainty.

Disadvantages: Since evidence values range from 0 to 1.0, we cannot distinguish between lack of belief and total disbelief.

4.1.4 Certainty Approach

Certainty theory [Shortliffe and Buchanan, 1975] is a theory developed for use in expert systems. It was developed in an attempt to overcome some of the problems associated with probability theory.

In certainty theory, a 'certainty measure' C(S) is associated with every 'factual' statement S such that [Frost 1988]:

a) C(S) = 1.0 if S is known to be true.

b) C(S) = -1.0 if S is known to be false.

c) C(S) = 0.0 if nothing is known about S.

d) Intermediate values indicate a measure of certainty or uncertainty in S.

Knowledge which shows how factual statements are related is represented as a set of rules such as:

* if S1 then S2 with certainty factor 0.8
* if S2 and S3 then S4 with certainty factor 0.5

The certainty factors associated with rules are measures of reliability of those rules. In general, rules are written with the following format:

if A then X with certainty factor CF

where A is called the condition part and X the conclusion. If the condition part is true, i.e., if it has a certainty value of 1.0, then that rule can be used to compute a new certainty value for its conclusion as follows:

a) If C(X) and CF both are positive, then new certainty of X, denoted by C(X|A) is computed by
$C(X|A) = C(X) + CF * [1.0 - C(X)]$
This equation can be explained as follows:
If the certainty value C(X) of a statement is positive, then the most that a rule with positive CF can increase the certainty of X is 1.0 - C(X). This amount is multiplied by CF and added to C(X).

b) If C(X) and CF both are negative, then
$C(X|A) = C(X) + CF * [1.0 + C(X)]$

c) If C(X) and CF are of opposite sign, then
$C(X|A) = [C(X) + CF] / [1.0 - MIN \{|C(X)|, |CF|\}]$

Advantages: This theory provides a means of manipulating the subjective estimates of certainty such that the calculated certainty values are intuitively appealing even if exact values of probabilities cannot be obtained.

a) The resulting certainty values lie between -1.0 and +1.0, so in this case the values -1.0 , 0.0 and +1.0 are well defined without any confusion and thus this overcomes the problem posed in the probability theory.

b) If two contradictory rules are applied in such a way that the certainty of one is equal to the certainty of the other, then their effects cancel out.

4.2 Building Blocks to Implement Non-binary Values

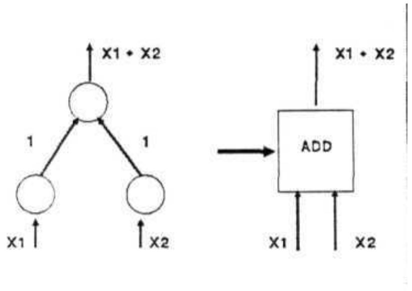
As seen above uncertainties cannot be absolute binary values. The grading is always relative and hence it is essential to deal with non-binary values.

As discussed in Chapter 3 so far no one has attempted to resolve this problem of utilizing non-binary values as inputs. This is essential for the present work. A methodology developed by us is presented in the following pages to utilize non-binary inputs.

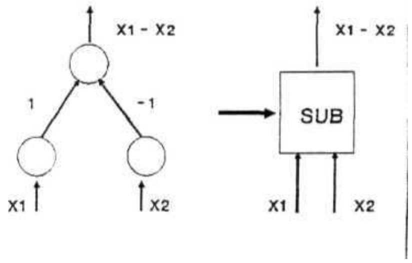
To enable a neural network to handle uncertainties in the form of non-binary inputs, some entirely new building blocks were formulated so as to produce a special node that will accept non-binary inputs.

These building blocks are:

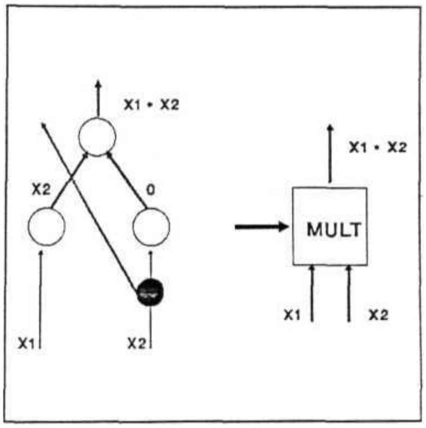
- Neural net that adds its two inputs x1 and x2, and outputs that sum as shown below.



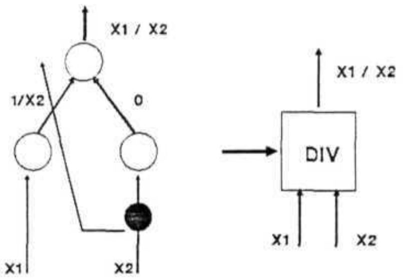
- Neural net that subtracts one input from another and outputs that result as shown below.



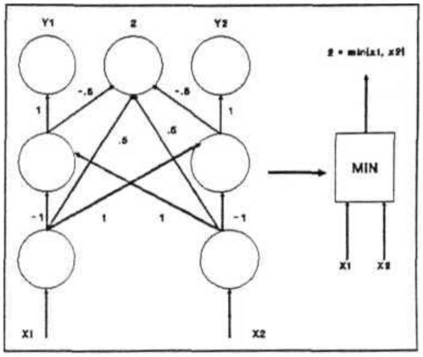
- Neural net that multiplies its two inputs and **outputs that** result as shown below.



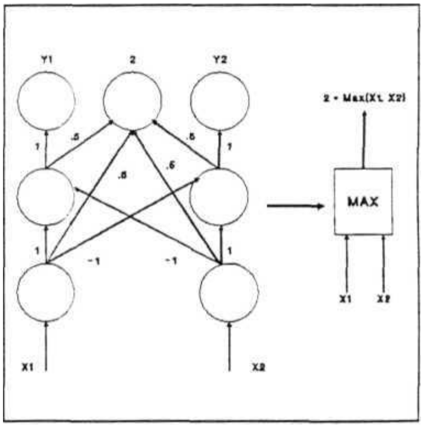
- Neural net that divides one input by another and outputs that result, as shown below.



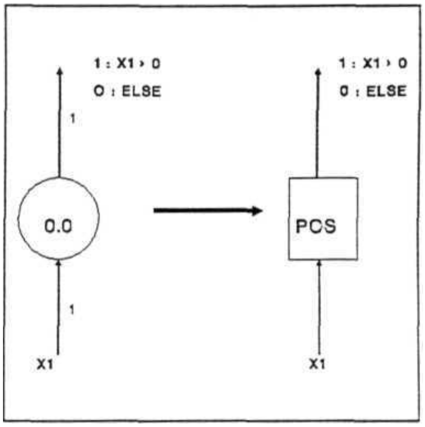
Neural net that outputs the minimum of two inputs as shown below.



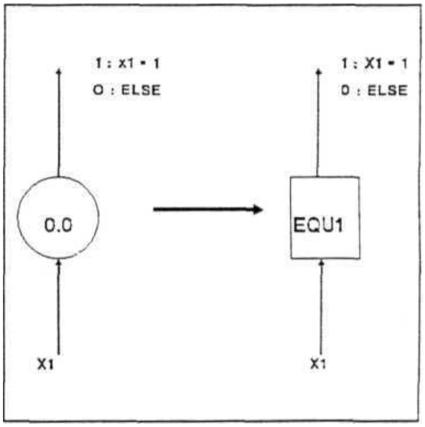
• Neural net that outputs the maximum of two inputs as shown below.



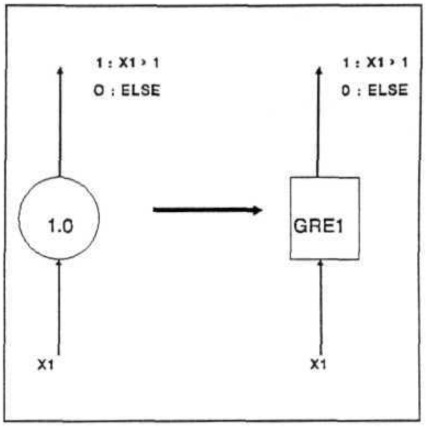
Neural net that outputs 1 if input is positive as shown below.



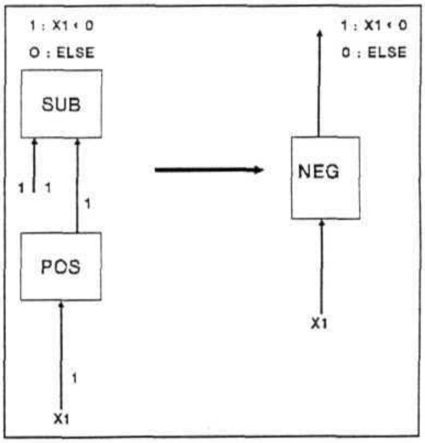
- Neural net that outputs 1 if input is 1 as **shown** below.



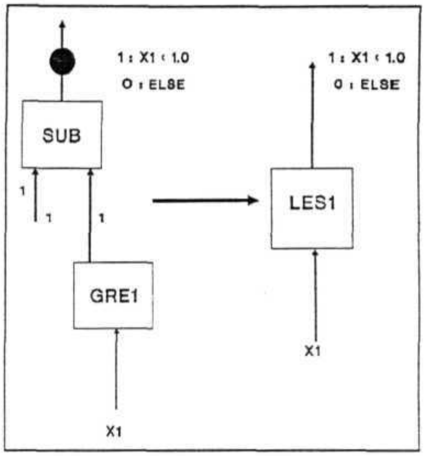
- Neural net that outputs 1 if input is greater than 1.0 as shown below.



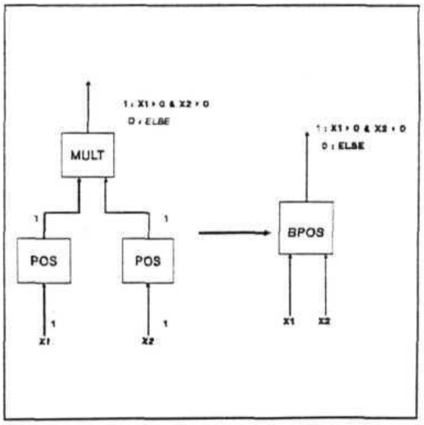
• Neural net that outputs 1 if input is negative as shown below.



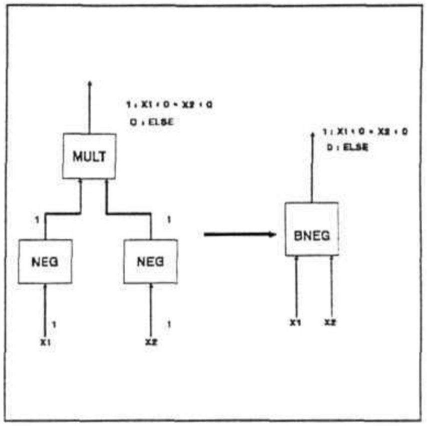
Neural net that outputs 1 if input is less than 1 as shown below.



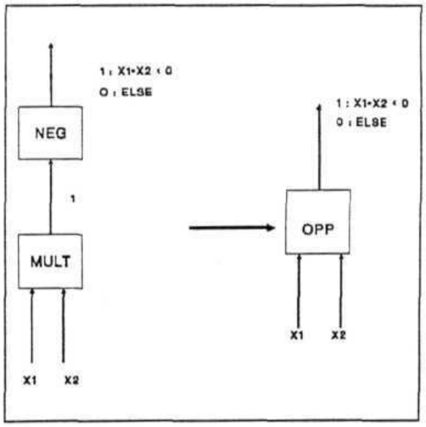
- Neural net that outputs 1 if both the inputs are positive as shown below.



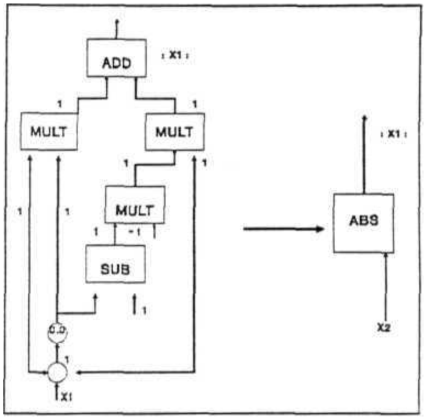
- Neural net that outputs 1 if both the inputs are negative as shown below.



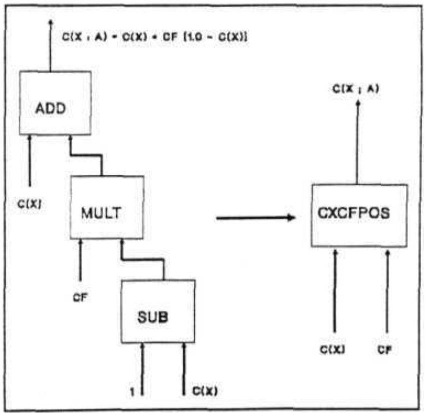
- Neural net that outputs 1 if the two inputs are of opposite signs as shown below.



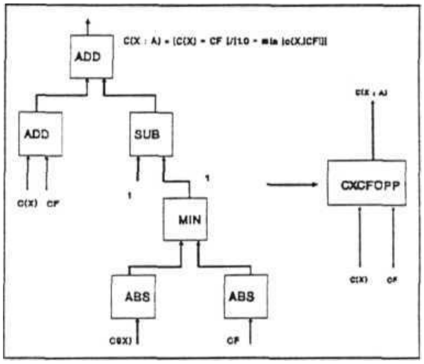
- Neural net that outputs the absolute value of the input as shown below.



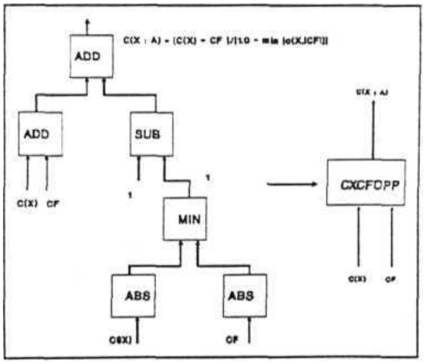
- Neural net that outputs modified values of X by taking initial value of conclusion C(X) and·rule strength (CF) as positive valued inputs by using the formula: C (X/A) = C(X) T C F * [1.0 - C(X)] as shown below.



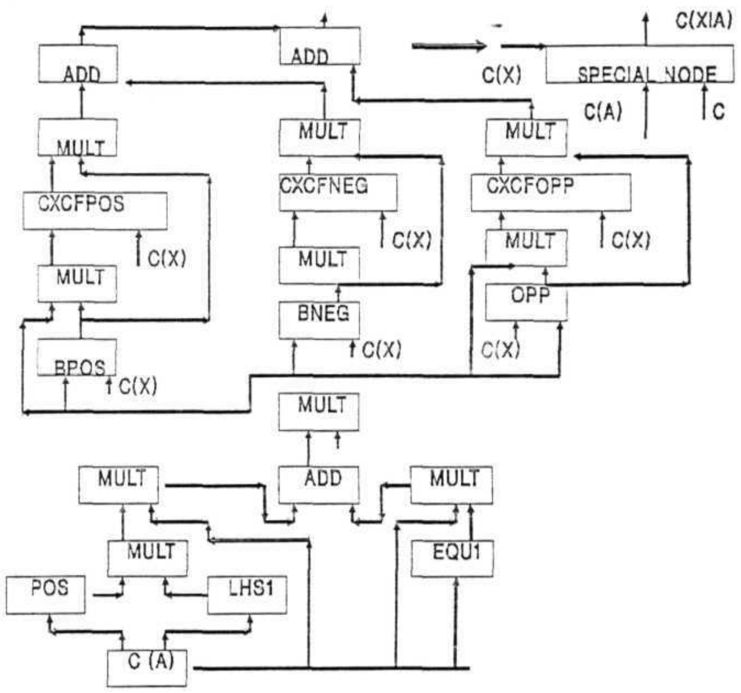
- Neural net that outputs modified values of X by taking initial value of conclusion C(X) and rule strength (CF) as negative valued inputs by using the formula: C(X/A) = C(X) + CF * [1.0 + C(X)] as shown **below.**



- Neural net that outputs modified value of X by taking initial value of conclusion C(X) and rule strength (CF) as opposite sign valued inputs by using the formula: C(X/A) = **[C(X)** + CF] / [1.0 - min{ | C(X) |, | CF | }] as given below.



- Special neural net developed to find out the modified premise strength of conclusion x and modified rule strength CF, by taking initial conditional strength A, initial rule strength CG and initial conclusion strength X. This handles certainty paradigm, as shown below.



4.3 Representation of Rule Format, Parsing and Rule-base

4.3.1 Rule Format

Rule is represented in a from of if-then-else statements. 'If' consists of condition part and an action part. If the condition part is satisfied, premises in action part are instantiated. Condition part and action part is separated by "→". A rule can consist of any number of premises. Operators allowed in condition part are (NOT) , (OR) and & (AND) . They inherit the properties of corresponding Boolean operation. Only one operation & (AND) is allowed in conclusion part.

Each rule should follow the following abstract format:

- Rule strength (any float value within -1.0 & + 1.0) followed by a blank.
- Condition part should consist of:
- Any character string (Premise) followed by a blank.
- Each premise should be followed by an operator and a blank.
- Action part follows the same format as condition part, following the operator restriction mentioned above.

Each rule is taken as a string; and is parsed by a rule parser (RP) to give the following:

- Rule strength
- No. of premises in condition part
- For each premise
  - Premise
  - Premise strength

- No. of operators
- List of operators
- No. of premises in action part
- For each premise
  - Premise
  - Premise strength

4.3.2 Rule Base Format

All the rule strengths and Premise strengths should be within the range of -1.0 to + 1.0.

User will be giving the Rule Ease in the following abstract format:

- No. of rules in the rule base.
- Each rule should consists of the following information
  - Rule strength
  - Conditional part
  - + Each premise should be followed by a blank, operator and blank.
  - Followed by a string "→"
  - Conclusion part
  - + Each premise is separated by a blank and an operator (only AND is allowed).
- No. of facts provided.
- Each premise (fact), its evidential strength separated by a blank.

Example:

```
4
0.98 B & C | D →F
0.83 F | H →K & I
0.88 B &L→ G
0.67 F ¦ G →X
```

5	
B	0.67
C	0.78
D	0.67
H	0.7
L	0.6

4.4 Evidential Reasoning: NN Approach

In the process of reasoning, each conclusion premise is taken as a neural node. Certainty factor of a rule is taken as a rule strength. The premises which are not involved in the conclusion part of any rule, in rule base, are considered as input node of a neural network. Premises that are in conclusion part of a rule are not specified in **condition** part of an other rule are taken as output nodes of a neural network. Rest of the premises are taken as intermediate/ hidden nodes. Premises whose certainty **strengths** (bias values) are not known, are taken as zero.

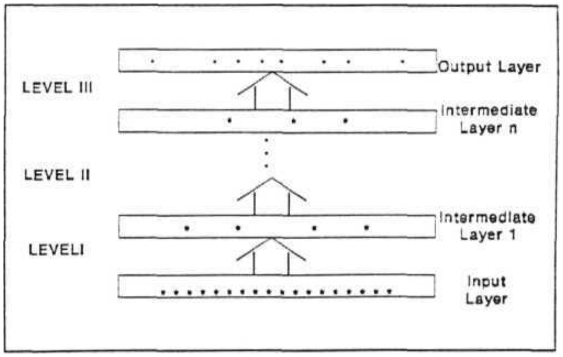


Figure *20. Structure of a neural* ***net.***

Structure of a neural network to be developed is shown in Figure 20. It consists of three different layers. Initially connections at level I will be established, by considering input node and hidden nodes of interr.ediate layer 1. Later, connections between intermediate layers are established according to the levels. Finally connection at level III between final intermediate layer and output layer is established.

Process:

a) Transform the rule base into a neural network by considering the procedure mentioned above.

b) Rule consists of any number of conditional or conclusional premises. Either AND *&(or OR (|) or NOT (") operators are allowed in conditional part. Only AND operator is allowed in action part.

c) If there is an AND operator in condition part, MINNET is invoked to calculate the modified certainty factor of the condition part.

d) If there is an OR operator, MAXNET is invoked.

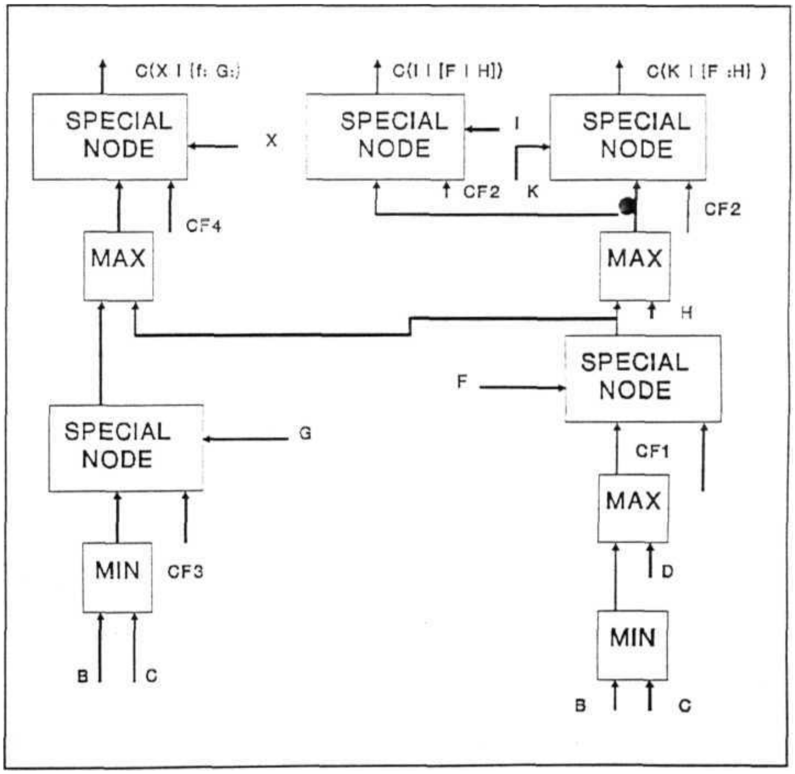
e) after processing the condition part, special node is invoked to handle certainty paradigm as specified above.

f) After developing the neural net, belief values are propagated through the NN.

g) During propagation, rule strength and premise strengths will be modified.

h) After propagation, if the neural net does not land up with the desired result, it should be trained. The process of training will be discussed in the next section.

- Example of the neural net developed, for the given rule base
  B&C | D -> ?
  F | H -> K&I
  B&L -> G
  F | G -> X
  as shown below



4.5 Training/Learning Factual Information

After propagating belief values through neural net, net will be reaching the final conclusion with some confidence values for final conclusions. If these confidence values are not the desired ones, then the net has to be trained accordingly to reach the desired values. Training will be done only for the final conclusion i.e., for the nodes at output layer. We are assuming the desired values are used to represent the knowledge which we want to embed in the network. (Before we train a network we must check whether there is a problem at the initial stage where a particular CF has been attached to a rule, can it be modified to get the desired resulr. After exploring these possibilities, if still required, we must proceed to train the NN) . The process of training is mentioned below:

Procedure:

Step 1 : Get the desired values for the final conclusions.

Step 2 : For each conclusion, calculate the error obtained (difference between desired & obtained values).

Step 3 : If the error is not zero, find the rule whose action part consists of this particular conclusion.

a) Modify the corresponding Rule strength.

$CF_{NEW} = (C_{DESIRED} - C_{OLD})/(1.0 - ABS(C_{OLD}))$

$CF_{MODIFIED} = CF_{NEW}/C$(Condition part)

If $CF_{MODIFIED}$ is not with in -1.0 & +1.0, then obtained CF is not valid one, else go to step 4

b) If $CF_{MODIFIED}$ is not valid, then modify the premise strengths of the condition part.

C(Condition part) CF_{NEW}/CF_{OLD}

If the condition premises are the conclusions for any other rule, then those should be modified recursively, until the condition satisfies. If the modified conditional strengths are not valid then go to step 3c else go to step 4.

c) Modify the initial conclusional strength

$C_{INITIAL}$ (Conclusion) = CF_{NEW} - CF) / (1.0 - CF)

If $C_{INITIAL}$ is not valid, then rule base is considered to have inconsistent factual information.

Step 4 : Repeat from step 2 for another conclusion.

Step 5 : Train the net until it reaches the stable state.

Step 6 : After reaching the stable state, if the desired values are obtained, then the net is said to be trained. Otherwise some conclusion should be drawn regarding this case.

4.6 Rule-base Consistency

Making a rule base consistent is a difficult problem [Wilkins and Buchanan 1986]. Existing approaches to this problem fall into several categories: Interaction with human experts [Davis 1976; Eshelman and McDermott 1986; Kahn et al. 1985], machile learning [Michalaski et al. 1983; Quinlan 1987], justification or explanation based on domain theory [Mitchell et al. 1986; Smith et al. 1985], empirical refinement [Ginsberg et al.

1988; Lee and Ray 1986; Politakis 1982; Rada 1985]. Attempts have been made by Hinton and Sejnowski [1983]; Geffner and Pearl [1987] and Fu [1991], to address this problem using NN approach. We think if the NN is not able to be trained to achieve the desired result by modifying the factual information provided, then the rule base given, is said to be inconsistent.

Inconsistency can arise either with the factual information provided or with the rule base itself. Now, the question arises how the inconsistency in rule base is defined? Is it in the same way as defined in the conventional expert systems? or Is it different?

Definition of inconsistency in conventional ES: Rule base is said to be inconsistent if it consists of conflicting rules. Conflict rules are those which succeed in the same situation but with conflicting conclusion.

Our definition of inconsistency: Rule base is said to be inconsistent if the NN (which corresponds to the rule base) is not able to train to give the desired values of some conclusions in the rule base.

Note: The desired values are representative of the tentative state of the knowledge about the domain. Training is merely an attempt to embed this knowledge into **the** network. Thus granted this set of desired values are true.

As this is beyond the scope of this thesis we will stop the discussion here.

Chapter 5

Equity Shares—An Analysis: A Neuro Knowledge Engineers Perspective

In this chapter we do an incisive analysis of equity shares. The objective of this analysis is to incorporate this knowledge in our framework. This would help us in developing our rule-base and would give us a deep insight into the realm of equity shares and related decision making.

5.1 Equity Shares

Equity investment is a variable-income investment option, whereas small saving schemes, bank deposits, company deposits and debentures, etc. are fixed-income investments. Whether a company does well or not, it has to pay interest on deposits. There is no such obligation to pay dividend on equity shares; it may be declared in certain years, be skipped in certain others. It could go up, cone down, or remain steady. Since equity capital is risk capital. If a company does well, the investors are benefited, otherwise not.

To make money from deposits and debentures one need not be as well-informed, knowledgeable—as an investor in equity shares.

5.2 Analysis of Equity Investment

Basically there are two methods of equity investment analysis for taking decisions like buying, selling and holding: fundamental analysis and technical analysis, Fundamental analysis is a value-based approach. Technical analysis is a market-based approach. [Yasaswy 1990]

Fundamental analysis is a value-based approach. This is a conservative and non-speculative approach to evaluate equity shares. Fundamental analysis consists of a three phase analysis: Economic analysis, industry analysis and company analysis. The perspective of a fundamental analysis is long-term. Various financial ratios are used as aids to decision making.

Technical analysis is a market based approach. It gives more importance to the technical aspects of the market, such as prices, price changes and trading volumes. The time perspective of it is short-term.

5.2.1 Fundamental Analysis

Fundamental analysis is time-honored and value-based approach, based on a careful assessment of the "fundamentals" of an economy, an industry and a company. A fundamental analyst is not unduly influenced by what is currently happening on a particular day in the stock exchange. He looks at the general economic situation, makes an evaluation of the particular industry and finally does an in-depth analysis - financial and non-financial of the specific company. Thus it is a 3-phase analysis - of economy, industry, and company as shown in Table 1.

5.2.1.1 Economic analysis

The stock market does not operate in a vacuum. It is an integral part of the total economy of a country.

To get an insight into the complexities of the stock market, an investor should develop a good sense of economic understanding and interpreting important economic indicators, with reference to their impact on stock markets.

Examples:

a) A favorable monsoon will have a positive impact on stock market. During years when the monsoon is good, Indian economy performs well with good growth rates in Gross National Product. As the purchasing power of the people goes up the aggregate demand goes up, and companies do well. Hence their profits go up and the investors are benefited.

Table 1. The Three Phase Fundamental Analysis.

Phase	Nature of analysis	Purpose	Tools & Techniques
First	Economic analysis	To assess the general economic situation in the country, its major trading partners, neighbors, etc.	Economic indicators—lead, lag and coincidental.
Second	Industry analysis	To review the prospects and problems of a specific industry and its segments.	Performance indicators, aggregate demand and supply position, internal and external competition, government policies.
Third	Company analysis	To analyse the financial and the non-financial aspects of a company and determine whether to buy/sell/hold shares of that company.	Non-financial aspects like promoter, management, product quality, corporate image, location, etc. Financial aspects like earnings per share, sales, profitability, dividend record, asset growth, etc.

b) Productivity of public sector enterprises like railways, coal, power, etc. play a crucial role in deciding the fate of our economy.

There are certain economic indicators which can be studied to assess the national economy as a whole. Some are known as leading indicators which foretell, in their own language, what is going to happen. Good examples of leading indicators are employment position, rainfall and agricultural production, fixed capital investment, corporate profits, money supply, credit position and index of equity share prices.

There are some coincidental indicators. Some examples of coincidental indicators are GNP (Gross National Product), Index of Industrial Production, money market, interest rates and reserve funds with the commercial banks.

Then there are some lagging indicators, which reveal what has already happened. Some examples of lagging indicators are large-scale unemployment, piled-up inventories, outstanding debt, interest rates of commercial loans, etc. While these indicators are useful, they are by no r.eans infallible. One must use them with caution. These indicators can be helpful in understanding the economic trends and may enable you to adjust your investment strategy suitably.

Table 2 summarises the impact of sor.e economic indicators on the stock market.

5.2.1.2 **Industry** analysis

The second phase of fundamental analysis consists of a detailed analysis of a specific industry; its characteristics, its past record, its present state and future prospects. The purpose of industry analysis is to identify those industries which are likely to grow in future, and to invest in the equity shares of companies selected from such industries.

Every industry (and every company in a given industry) usually goes through a life-cycle with four distinct phases: (a) pioneering stage, (b) expansion stage, (c) stagnation stage, and (d) declining stage. An investor would be benefited by investing in an industry only in its pioneering expansion stages. One should quickly get out of industries

Table 2. Economic indicators and their impact on the stock market.

Indicators	Favorable Impact	Unfavorable Impact
Gross National Product	High growth rate	Slow growth rate
General employment position	Full or nearfull employment	Underemployment and unemployment
Domestic savings rate		Low
	Low	High
Tax rates	Low	High
Foreign exchange rates	High	Low
Balance of trade	Positive	Negative
Deficit financing	LOW	High
Inflation	LOU	High
Agricultural production	High	Low
Industrial production	High	Low
Power supply	High	Low
Freight movement of railways	High	Lou
New house construction	High	Low

which have reached the stagnation stage, and before they lapse into decline. The particular phase of an industry can be understood in terns of its sales (volume and value) and profitability. [Aggarwal 1985]

Industries which my be doing well today may in future, face stagnation and decline as a result of changes in social habits (e.g., the cigarette industry is bound to suffer with increasing emphasis on the health hazards of smoking), or from changes in statutory controls (e.g., the prohibition), or from the emergence of excess capacity and consequent cut-throat competition (e.g.,polyester), or as a result of rising prices (e.g., the Indian refrigerator and air-conditioning industries are outpriced for vast segments of the domestic market largely from heavy excise imports). For an investor, these analytical insights into the various industries are necessary.

The method of evaluation of the Industry should encompass four critical areas:

(i) What are the strength of the industry ?
(ii) What are its vulnerabilities ?
(iii) What are the opportunities available to it?
(iv) What are the threats faced by it ?

Such a comprehensive analysis is not going to be a simple exercise. The investor should evaluate the industry with the help of financial and non-financial data he may have access to.

5.2.1.3 Company analysis

Many investors find that though a particular industry may be doing very **well,** certain companies in that industry may not be in good shape. Hence selecting the individual companies for investment in a given industry is equally important.

There are two major components of company analysis Financial and non-financial. A good analyst tries to give balanced weightage to both these aspects. Overemphasis on either may lead to a distorted analysis.

Non-financial aspects: Many non-financial aspects of the company should be evaluated by an investor. The non-financial factors are listed in Table 3.

Table 3. A framework for General (Non-financial) Analysis of Companies.

Aspect	Review Questions
History, promoters and management	How old is the company? Who are the promoters? Is it family managed or professionally managed? What is the public image and reputation of the company, its promoters and its products?
Technology, facilities and production	Does the company use relevant technology? Is there any foreign collaboration? Where is the Unit located? Are the production facilities well balanced? Is the size the right economic size? What are the production trends? What is the raw material position? Is the process power-intensive? Are there adequate arrangements?
Product range, marketing selling and distribution	What is the company's product range? Are there any cash cows among the products? How effective is the market network? What is the brand image of the products? What is the market share enjoyed by the products in the relevant segments? What are the effects and costs of sales promotion and distribution?

Industrial relations, productivity and personnel	How important is the labor component? What is the worker productivity? How is the labor situation in general?
Environment	Are there any statutory controls on production, price, distribution, raw materials etc.? Are there any major legal constraints?

History of the company, Promoters and Managements, Technology, Production, Marketing, Environment (statutory controls), Industrial Relations and Sales, Personnel etc.

Equity analyst attaches great importance to the following ratios in financial analysis:

Earning per **share (EPS)**: This indicates the post-tax profits earned per share. The higher the better.

EPS = profit after tax /No. of equity shares.

Price-earning ratio (P/E ratio): This ratio indicates the relationship between the market price of the share and the earnings per share. Whether a particular company's P/E ratio is high or low may be understood with reference to the All-Industry average, and also with reference to the specific average.

Price-Earning Ratio = Market Price of the share / Earnings per share

Book value per share: This ratio indicates the asset-backing available of each share. The higher, the better.

Book value per share = Shareholders' funds + reserves / No. of Equity Shares

Return on net worth = this indicates the post-tax return on the shareholder's funds. The higher the better.

Return on net worth = Profit after tax / Shareholders' funds X 100

Dividend cover: This indicates the extent to which the equity dividends are protected by the earnings. The higher the better.

Earnings per share / Dividend per share

Profitability of sales: This indicates the profitability or otherwise of the sales. The higher this ratio, the better the profitability.

Profitability of sales = Profit before tax / sales X 100

Debt-equity ratio: Debt (i.e.,loans) is measured as a percentage of equity (i.e., the shareholders' funds). The lower the ratio, the better.

Debt-Equity ratio = Loans / Shareholders' funds X 100

5.3 Technical Analysis

Technical analysis deals with the factors of supply of, and demand for shares. Technical analysis is market oriented. A true technical analyst is not worried about the company's assets, turnover, dividends, reserves, product, or even its name. He looks only at the market situation for the company to decide about investing in it.

According to the technical analyst all such relevant factors, which affect the market, get reflected in the volume of stock exchange transaction, and the level of share prices.

The basic assumptions underlying technical analysis are:

a) Market value is determined solely by the interaction of supply of, and demand for shares.

b) Supply and demand are governed by **many** rational and irrational factors.

A technical analyst game plans are sir.ple:

- If the market price is raising, BUY.
- If the market price is going down, SELL.
- If the market price is steady, WAIT.

Some technical indicators:

There are many tools and indicators used to understand and interpret the market position as a whole, and also individual scripts. Some of them are:

Market averages: The patterns of the market averages like BSE (Bombay Stock Exchange) sensitive index, BSE National index, etc, are studied to obtain clues for future action.

Trading volume: The figure relating to trading volumes are studied to assess the price pressure created by high trading volume or low trading volume. If price rise is accompanied by a trading volume, it is sure sign of upsurge in demand.

c) Short interest: This indicates the total number of shares sold short. Short selling takes place when prices are expected tc decline in a later period. When the market booms, short selling diminishes.

d) Irregular prediction tools: There are sor.e other methods which are used for predicting the market. They are understood as irregular prediction tools. These include the following:

- Fricas gc up on Friday because all leading open-ended mutual funds calculate Net Asset Value as on Friday.
- Prices usually go down in February every year from budget fears.
- Prices are pulled down on March 31 every year as that being the valuation date for Wealth tax purpose.
- Prices ara depressed before general elections.

Cr.a has tc go through a great deal of trial and error before he can develop reasonable interpretative skills.

To arrive at accurate results one has to do the fundamental as well the technical analysis.

5.4 Advantages of Equity Investment

There are many advantages of investing in equity shares of well managed and successful companies. The most important of these are:

5.4.1 Capital Appreciation

Equity shares of good companies appreciate in value and act as a partial hedge against inflation. Consequently the purchasing power of your investment in such shares is generally protected to a great extent.

Bonus shares: Successful companies frequently issue bonus shares, subject to guidelines issued by the Controller of Capital Issues from time to time. After bonus shares are issued, shareholders are entitled to dividends not only on the original shares but on the bonus shares as **well.**

Annual dividends: All reasonably profitable companies try to maintain a steady rate of dividends. In fact, many of them declare interim dividends as well.

Rights shares: When a company wants to issue new equity shares, these must be first offered to the existing shareholders on a pro-rata basis, unless the existing shareholders agree to give up this right. Such shares are known as "Rights Shares". This right to further shares can be sold in the market. So shareholders who do not wish to subscribe for further shares can sell their rights at a profit, if there is a good demand for them.

Voting rights: As an owner of the company, an equity shareholder, enjoys voting rights in the general meeting of the company.

As a pledge: Equity shares of the selected companies can be pledged as security to raise loans from banks and other financial institutions.

Tax benefits: Under section SOL of the Income Tax Act, dividends on shares of Indian companies are exempt from income tax up to Rs. 10,000 per year. Also tax relief is available for investments in certain new company shares under Section 80CC/88A of the Income Tax Act.

Marketability: Listed equity shares which are actively traded and quoted on stock exchanqes can be sold without difficulty. Whenever you want money, you can ring up your broker and dispose of the shares at or around the prevailing market prices. There are now 19 stock exchanges in India: Ahmedabad, Bangalore, Baroda, Bhubaneshwar, Bombay, Calcutta, Cochin, Delhi, Gauhati, Hyderabad, Indore, Jaipur, Kanpur, Ludhiana, Madras, Mangalore, Patna, **?une** and Rajkot. However not all the shares which are listed are actively traded.

While all these advantages are tempting, there are some attendant problems as well.

5.4.2 Problems of Investing in Equity Shares

Changing market values: The market values of actively traded **equity** shares seldom remain constant. They keep fluctuating; some moderately, but other **violently.** These fluctuations in the market prices are likely to cause anxiety and discomfort for the amateur investor.

Need for constant watch: Equity investment is not a one-shot affair, it demands your continuing involvement. You have to keep constant: watch on the environmental factors sUch as the industry's prospects, the company's performance, etc. Some times it can be more preoccupying than a fulltime job.

Criticability of timing: Since timing is critical both while buying or selling shares you have always to be alert. If you miss a good right opportunity once, you nay have to wait for a long time for the next one.

Uncertainty of government policies: Consistency has never been a strong point with our government, Uncertain changing policies of the policies of companies, which **in turn,** affects the share holders. A change in the government of course leads to considerable changes in policies.

5.5 A Framework for Analysis and Decisions

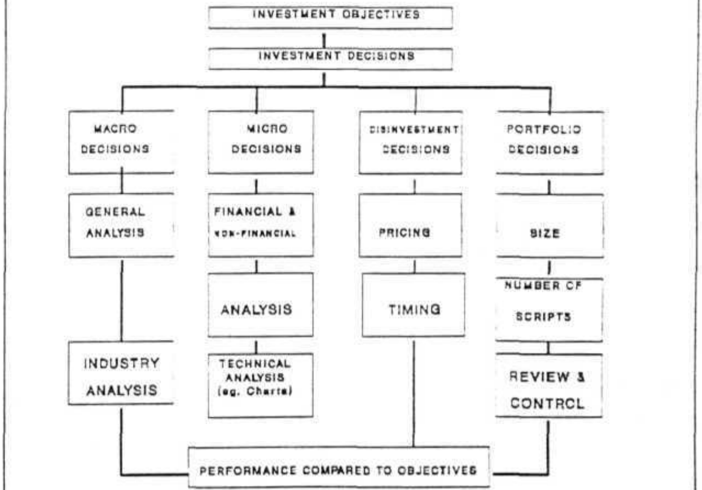


FIGURE 1. INVESTMENT ANALYSIS AND DECISIONS - A FRAMEWORK

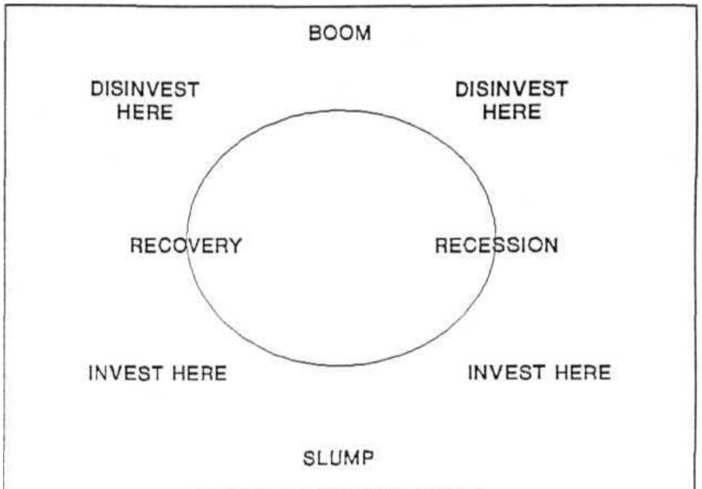


FIGURE 2. BUSINESS CYCLE

Let us understand the type of decisions you have to make in equity investments.

5.5.1 Macro Decisions

a) Whether it is an opportune time for equity investment. There are booms and slur.ps in the market. Ideally or.e should invest at the end of a slump and quit before the end of a boom, as shown in the Figure 2.

To understand these stock market cycles, which are interlinked with the business cycles, an appreciation of important macro-economic factors which have a bearing on the stock market is necessary.

b) Whether a particular industry (or industries) is right for investment or not at a given point of time. There are certain sunrise industries (e.g., electronics) and some sunset industries (e.g., Jute) . There are high-tech industries (e.g., instrumentation) and low-tech industries (e.g., solvent extraction) . There are capital intensive industries (e.g., petrochemicals) and labour intensive industries (e.g. textiles). Thus industries can be classified into several types. Each industry goes through a certain life cycle from a small beginning to r.assive growth to stagnation to eventual decline.

A good insight into these broad industry aspects, will help choose the right type of industry for investment at an opportune time.

5.5.2 Micro Decisions

The micro decisions relating to equity investments deal with three issues whether one is talking about existing companies or **new** ones.

a) Selection of a specific company: All companies—in any particular industry you choose for investment—are obviously not equally good. You have to choose the right company or companies based both on financial criteria (based on balance sheet analysis and also non-financial reasons, such as management reputation, past track record, future plans, etc.

b) Deciding on the right price: Having chosen a company, you then need to decide whether its share is attractive at the prevailing price. Is it overpriced, Is it underpriced, or Is the price just right? Whether a price is right or not essentially depends not so much on a company's asset base but upon its earning power. If the earnings are good, and growing, a high price may be justified; otherwise not.

c) **Deciding on the right time:** The next aspect is the timing of your investment; i.e., Is it the right time to buy a particular share: In a way, the time and price issues are interlinked and may be examined together. A good understanding of charts of share price movement may help in timing your purchase. Obviously, one should not buy if the price is likely to go down in the near future; similarly, one should not sell if the price is likely to go up.

5.5.3 Disinvestment Decisions

Stock market profits are illusory until you sell your shares and book the gains. You must disinvest periodically. The disinvestment process is the mirror-image of an investment decision. All you have to do is to apply all the investment principles in reverse.

5.5.4 Portfolio Decisions

Prudent investors never put all their money in just one or **two** shares, because the risk of such a concentration is too high. On the other hand, if you diversify too much, the average performance of your portfolio will be mediocre. Hence you have to strike a proper balance between non-diversification and excessive diversification. Also, the portfolio should be reviewed periodically, shuffled whenever necessary, and otherwise properly managed.

We will now apply the investment analysis and decision paradigm (Figure 1) on equity analysis of cement industry.

5.6 An Example

It is amply clear from the discussion presented above that decision making in this activity is a very complicated process. To aid the decision maker help of computers is necessary. Over present system that draws from the latest advances in AI such as NN is expected to make the decision making process more effective.

To illustrate the performance of our system, cement industry has been chosen. We have analyzed different aspects of cement industry based on various methods enumerated earlier. We have derived rules. Information about six leading cement companies, ACC Cement, Ramco Cement, Rayalseema Cement, Panyam Cement, Kakatiya Cement and Coramandal Cement has been fed into the system. Figures 3 and 4 illustrate in a capsulated form different types of analysis used in formulating the rules.

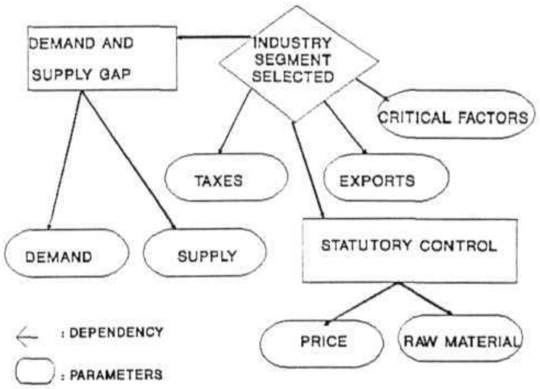


Figure 3. Selecting the industry segment.

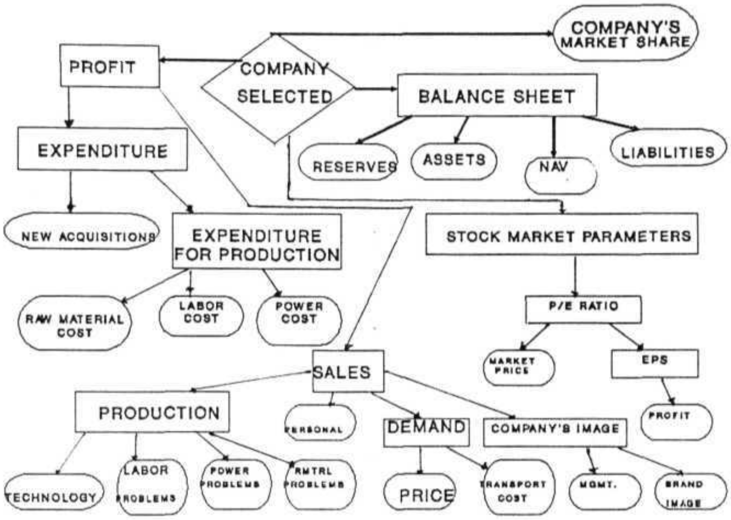


Figure 4. Selecting the company.

The following parameters were considered for the analysis of the above cement companies.

Stock market parameters:

a) P.E. ratio
b) Market price
c) Earning per share
d) Profit after taxes

Balance sheet:

a) Reserves
b) Assets
c) Net asset value (NAV)
d) Liabilities
e) Promoter's contribution

Expenditure for production:

a) Raw material cost
b) Labour cost
c) Power **cost**

Production problems:

a) Technology related problems
b) Labour problems
c) Power problems
d) Raw material problems

Demand for the product:

a) Brand image
b) Price
c) Transport Cost

Company's image:

a) Management
b) Brand Image

Based on the analysis a Knowledge Base was built. Some sample rules are given below:

Rules to Grade Production

```
rulep1:

if company has labor problems
     and raw_material problems are intense or they exist or do
     not exist
     and power problem are intense or exist or do not exist
     and company's technology advances are high
then
     company's production is high

rulep2:

if company has labor problems
     and raw material problems are intense or they exist or do
     not exist
     and power problem are intense or exist or do not exist
     and company's technology advances are normal
then
     company's production is low

rulep3:

if company has no labor problems
     and has no raw_material problems
     and has no power problems
     and company's technology advances are normal
then
     company's production is high
```

```
rulep4:
if company has no labor problems
     and has no raw_material problems
     and has no power problems
     and company's technology advances are high
then
     company's production is very high

rulep5:
if company has no labor problems
     and has no raw_material problems
     and power problems are intense
     and company's technology advances are high
then
     company's production is high

rulep6:
if company has no labor problems
     and has ok raw material problems
     and power problem are intense or exist or do not exist
     and company's technology advances are either high or
     normal
then
     company's production is ok

rulep7:
if company has intense or normal labor problems or does not
have labor problems
     and intense raw_material problems
     and power problem are intense or exist or do not exist
     and company's technology advances are high or normal
then
     company's production is low
```

```
rulep8:
if company has labor problems
     and has normal raw_material problems
     and power problem are intense or exist or do not exist
     and company's technology advances are high
then
     company's production is ok

rulep9:
if company has labor problems
     and has normal raw_material problems
     and power problem are intense or exist or do not exist
     and company's technology advances are normal
then
     company's production is low

rulep10:
if company has no labor problems
     and has normal raw_material problems
     and has intense or normal power probler.s
     and company's technology advances are high or normal
then
     company's production is low
```

Chapter 6

Inside Neuro Expert: Implementation and Results

This chapter discusses the design and implementation details of Neuro-Expert (Figure 1).

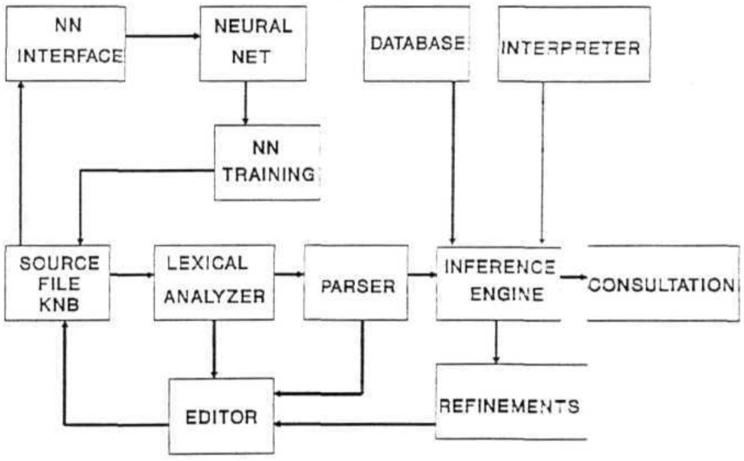


Figure 1. Neruo-Expert block diagram.

In this chapter, we set the stage with an insight into the system architecture and higher level modules. In the first section we look at the system from the designer's perspective. We shall dissect the system and take an in-depth look at each component of the system. This section also presents the main block-diagram and data flow diagram. We **will** briefly discuss the major modules of our system and explain how mapping facility has been developed to map rules from knowledge base to an Artificial Neural Network.

In the second section we look at the system from programmer's perspective. We touch upon the syntactical interfaces and the functions provided by Knowledge Definition Language (KDL) . The section ends with scr.e sample rules from the Shares Knowledge base. This knowledge base has more than 200 rules. We are presenting here scr.e sample rules to indicate our knowledge engineering methodology.

In the third section, we take a look at the system from a user's perspective. Issues included here are, how easy it is to use and learn the system. This section describes the user interface in detail.

6.1 System Design

The system (Figure 2) can be decomposed into the following major modules (Figure 3):

- Lexical Analyzer
- Inference Engine
- Interpreter
- Database Interface
- Knowledge Base

- User Interface
- NN Interface
- NN Training

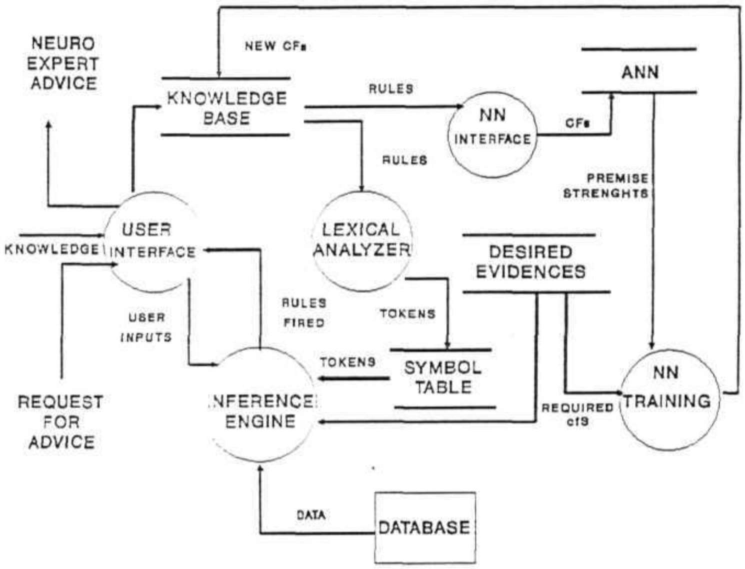


Figure 2. Level-0 data flow diagram for Neuro-Expert.

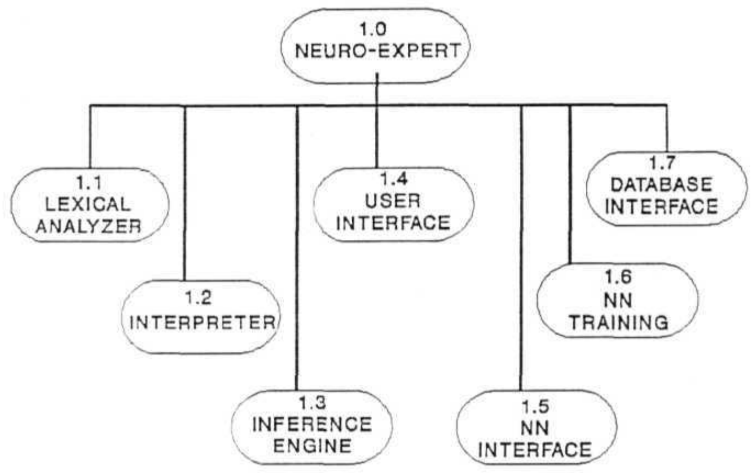


Figure 3. Functional decomposition diagram for Neuro-Expert.

6.1.1 Lexical Analyzer

The lexical analyzer (Figure 4) is an important module of the system that loads the knowledge base into memory and analyses it for both syntactic and semantic errors. This module refines the knowledge base by removing comment lines and unused blank lines. The lexical analyzer has been developed after a careful design of the language **structure.**

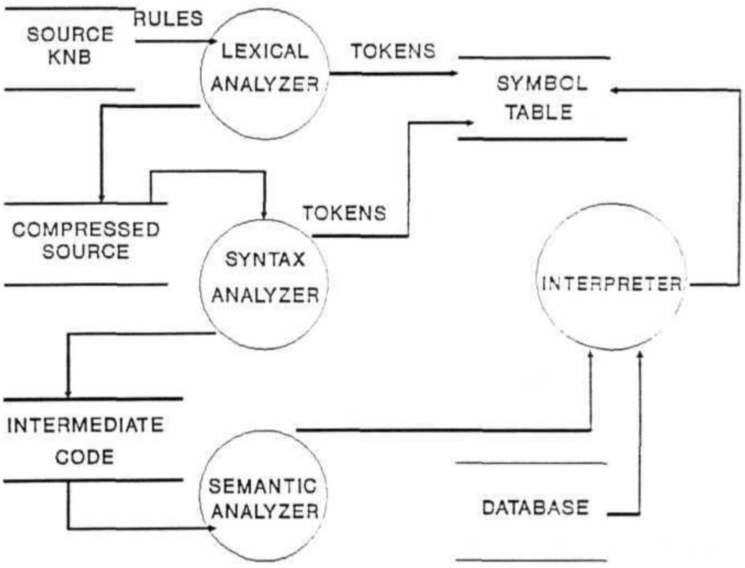


Figure 4. Data flow diagram for Lexical-Analyzer.

6.1.2 Inference Engine

The Inference Engine (Figure 5) consists of 3 algorithms for match-rules, select-rules and execute-rules and directing supports forward chaining with natch rules matching the condition elements against invoking memory, select rules choosing one dominant rule and execute-rules firing the rules by executing the actions of Right Hand Side (RHS) sequence while the inference engine is a forward chaining engine, backward chaining problem-solving strategies can be implemented.

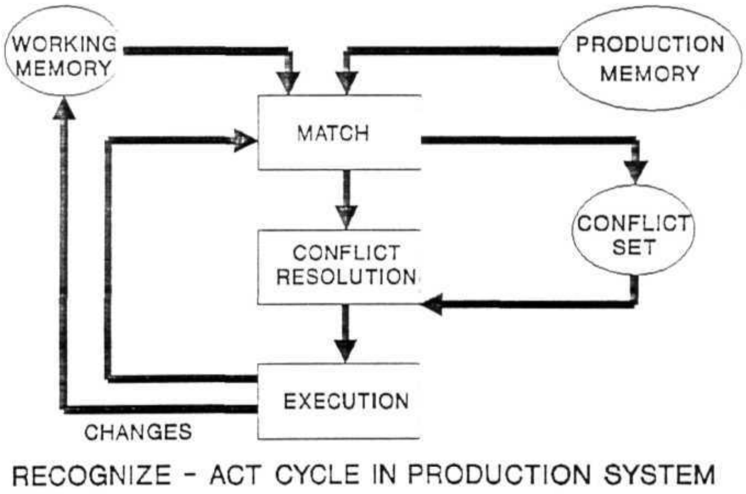


Figure 5. Block diagram for Inference Engine,

6.1.3 Interpreter

The interpreter is invoked only by the Inference engine. It analyzes statements in sequence and passes the result back to **the** Inference Engine, which in turn makes use of this result to take decisions and appropriate actions.

6.1.4 Knowledge Base

The Knowledge Base is an ASCII text file. The first step in creating acknowledge base is the acquisition of the expertise of experts from various sources such as text books, journals, heuristics etc. The knowledge thus obtained is to be coded in a form specified by the grammar rules of the Knowledge Definition Language (KDL). The Knowledge Definition Language is discussed in detail in Section 6.2.

6.1.5 Database Interface

Prime importance is given for the response time of rhe system. As a first step maximum care is taken to **avoid** asking unnecessary questions to the user. The data required by the system can be kept in a database file.

A database interface (Foxpro) is integrated with the system. The database filename and key field (to be used in locating a record) should be **known.** After reading a record **the** field names and their corresponding values will be available in the symbol table, so that later the field names **can** be considered to be user defined variables.

The design of the system has been kept open. Database interface for other popular RDBMS's like Oracle, Ingresetc., also could be developed and connected to Neuro Expert.

6.1.6 The User Interface

The Shell has an extremely user friendly interface. The various features of user interface include:

(a) Menus: With the user friendly interface provided with shell, it is easy to develop and use expert systems. The pulldown and popup menus enable the user to operate the system without having to refer to any other user guides of the software.

(b) Context-sensitive help: The system provides a context-sensitive help facility. If the help is invoked from the main menu, an index for help will be displayed, from which items can be selected using the cursor control keys. Pressing F5 from any help screen will take you back to the help index. The Up and Down arrow keys can be used to view through the help displayed.

The indicators at the top and bottom corners of the right side of the help window show the availability of more pages. The help can be activated at any point of time (at any menu).

(c) Integrated editor: To invoke the editor, choose Edit from the main menu. The user can edit either a knowledge base file or non-knowledge base file. The types can be selected using the Edit Sub menu.

When editing is over, the main menu will be restored. If the last edited file is a knowledge base file, then it will be automatically loaded into the system work area. At the present form of the system "Norton editor" is used for editing.

6.1.7 NN Interface

This module performs the function of mapping the rules **in** the knowledge base into the corresponding neural net. In the process of reasoning, each premise is taken as a neural node. Certainty factor of a rule is take as a rule strength. The premises which are not involved in the conclusion part of **any** rule, in rule base, are considered as input node of **a** neural network. Premises that are in conclusion part of a rule are not specified in condition part of another rule are taken as output nodes of a neural network. Rest of the premises are taken as intermediate/ hidden nodes. Premises whose certainty strengths (bias values) are not known, are taken as zero.

6.1.7.1 Sample file formats

All the Rule strengths and Premise strengths should be within the range of -1.0 & +1.0

6.1.7.1.1 User file format (Rules)

User will be giving the filename which consists of rules. This file would have an extension ".knb". Format of that file should be:

Ex:

```
    (#rule 1:
        (BC (0.87)) & (DE (0.78))
            → (G & FG) (0.98))

    (#rule 2:
        (~G) | (BC (0.67)) & (FG (0.87))
            → (X) (0.83))
```

6.1.7.1.2 User file format (Desired Evidences)

This file includes the information about the desired evidences. This file would have an extension ".evi". It should include

- No. of evidences
- Each premise and its desired evidence (separated by a blank)

Ex:

```
3
G 0.9

LI 0.99

YG 1.0
```

6.1.7.1.3 **Internal** file format I

This is an internal file used by the program to store the modified rule forms of the given rule base in the user file name. This file would have an extension ".dat".

It consists of eight attributes.

(i)	Rule Strength	:	float value
(ii)	No. of Conclusions	:	int value
(iii)	Each conclusion (premise)	:	char string
(iv)	Each premise strength	:	float value
(v)	No. of conditions	:	int value
(vi)	List of operators	:	char array
(vii)	Each condition (premise)	:	char string
(viii)	Condition strength	:	float value

"Shares.dat" for the example given above is

```
0.98 2 G 0 FG 0.67 & BC 0.67 DE 0.78

0.83 1 X 0 3 | & G 0 BC 0.67 FG 0.67

0.88 2 YG 0 E 0 3 & | X 0 DE 0.78 G 0

0.67 1 LI 0 2 | X 0 DF 0.7
```

6.1.7.1.4 Internal file format II

It consists of the neural net developed for the given rule base which consists of operation (function) followed by parameters. This file will have an extension ".net".

Ex: Neural net developed for the given rule base

```
MIN BC DE
SN 0 G
SN P FG
MAX G BC
MIN + FG
SN 1 X
MIN X DE
MAX + G
SN 2 YG
SN 2 E
MAX X DF
SN 3 LI
```

6.1.7.2 Neural net format

Example is given in the format of "shares.net". Information in the net will indicate the following features: (p1 & p2 indicates premise1 and premise2).

1. For each condition part (which consists of more than one premise)

   a) If the operation between first two premises is

      . '&' : MIN p1 p2 is written in the file

      . '|' : MAX p1 p2 is written in the file

   b) For the rest of the premises (if they are more then two)

      • If the operation is

      . '&' : MIN + p1 is written in the file

      . '|' : MAX + p2 is written in the file

2) If condition part consists of one premise

   CNE Rule no. p1 is written in the file

3) After condition part is over, special node is invoked to calculate modified premise strength for each conclusion. For each conclusional premise, the following statement is written in a file.
   SN Ruleno. p1 is written in the file

6.1.7.3 Special nodes developed

- ADD(xl,x2) – Neural net that adds its two inputs x1 and x2, and outputs that sum.
- SUB(xl,x2) – Neural net that subtracts one input from another and outputs that result.
- MULT(x1,x2) – Neural net that multiplies its two inputs and outputs that result.
- DIV(xl,x2) – Neural net that divides one input by another and outputs that result.
- MIN(x1,x2) – Neural net that outputs the minimum of x1 & X2 .
- MAX(x1,x2) – Neural net that outputs the maximum of xl & x2 .
- POS(xl) – Neural net that outputs 1 if xl >= 0 else it outputs 0.
- EQU1(x1) – Neural net that outputs 1 if xl = 1.0 else 0.
- GREl(xl) – Neural net that outputs 1 if xl > 1.0 else 0.
- NEGl(xl) – Neural net that outputs if xl < 0 else 0.
- LESl(xl) – Neural net that outputs if xl < 0 else 0.
- BPOS(x1,x2) – Neural net that outputs 1 if xl > 0 and x2 > 0 else it outputs 0.

- BNEG(x1,x2) – Neural net that outputs 1 if x1 < 0 **and** x2 < 0 else it outputs 0.

- 0PP(xl,x2) – Neural net that outputs 1 if xl * x2 are of opposite sign else it outputs 0.

- ABS(xl) – Neural net that outputs the absolute value of x1.

- CXCFPOS (CS,CF) – Neural net that outputs modified values of X by taking initial value of conclusion (CX) and rule strength (CF) as positive valued inputs by using the below formula.

  $C(X|A) = C(X) + CF * [1.0 + C(X)]$

- CXCFNEG(CX,CF) – Neural net that ouputs modified values of X by taking initial value of conclusion (CX) and rule strength (CF) as negative valued inputs by using the below formula.

  $C(X|A) = C(X) + CF [1.0 + C(X)]$

- CXCFOPP(CX,CF) – Neural net that outputs modified value of X by taking initial value of X and rule strength (CF) as opposite sign valued inputs.

  $C(X|A) = [C(X) + CF] / [1.0 - nin \{|C(X)|, |CF|\}]$

- NODEA(A,CF,X) – Special neural net developed to find out the modified premise strength of conclusion X and modified Rule strength CF, by taking initial conditional strength X. This handles certainty paradigm.

6.1.8 NN Training

After propagating belief values through neural net, net will be reaching the final conclusion with some confidence values for final conclusions. If those confidence values are not the desired one, then the net **has to** be trained accordingly to reach the desired values. Training will be **done only for** the final conclusion, i.e., for the nodes at output layer. We are assuming the desired values are used to represent the knowledge which we want to embed in the network. The process of training was discussed in Chapter 4. Examples on training are discussed now.

Some Examples:

Example 1: Trained

```
Give the file name which consists the Rule base
two.dat

INPUT TO THE PROGRAM
No. of rules in the given file = 4
rule [  0] is 0.9 B & C -> F & G
rule [  1] is 0.67 C -> D
rule [  2] is 0.8 D -> F
rule [  3] is 0.78 B -> C

evidences b = 0.400000

RULE STRENGHTS
CFRS[  0] = 0.900000
CFRS[  1] = 0.670000
CFRS[  2] = 0.870000
CFRS[  3] = 0.780000
```

```
PREMISE STRENGTHS
VARI =   F  EVIDI = 0.000000 EVID2 = 0.000000
VARI =   G  EVID1 = 0.000000 EVID2 = 0.000000
VARI =   B  EVIDI = 0.400000 EVID2 = 0.000000
VARI =   C  EVIDI = 0.000000 EVID2 = 0.000000
VARI =   D  EVIDI = 0.000000 EVID2 = 0.000000
VARI =  "F  EVIDI = 0.000000 EVID2 = 0.000000

NEURAL NET DEVELOPED
ONE  3 B
SN  3 C
MIN B C
SN  0 F
SN  0 G
ONE  1 C
SN  1 D
ONE  2 D
SN  2 "F

After PROPAGATION values are

RULE strengths
CFRS1 = 0.900000 CFRS2 = 0.280800
CFRS1 = 0.670000 CFRS2 = 0.209040
CFRS1 = 0.870000 CFRS2 - 0.181865
CFRS1 = 0.780000 CFRS2 = 0.312000

PREMISE strengths
VAR = F  EVIDI = 0.000000 EVID2 = 0.229732
VAR = G  EVIDI = 0.000000 EVID2 = 0.280800
VAR = B  EVIDI = 0.400000 EVID2 = 0.400000
VAR = C  EVIDI = 0.000000 EVID2 = 0.312000
VAR = D  EVIDI = 0.000000 EVID2 = 0.209040
VAR = ~F EVIDI = 0.000000 EVID2 = 0.770268
```

```
GIVE THE FILENAME WHICH CONSISTS OF DESIRED EVIDENCES
file name = two.evid
Reading desired evidences NS = 1
CAR[  0] = G NEW[  0] = 0.900000

SYSTEM IS ABLE TO TRAIN THE NET TO GET THE DESIRED VALUES
VALUES WILL BE SHOWN

FINAL MODIFIED VALUES ARE
VARI[  0] =  F EVID1 = 0.000000 EVID2 = 0.153131
VARI[  1] =  G EVID1 = 0.860957 EVID2 = 0.900000
VARI[  2] =  B EVID1 = 1.000000 EVID2 = 1.000000
VARI[  3] =  C EVID1 = 0.000000 EVID2 = 0.780000
VARI[  4] =  D EVID1 = 0.000000 EVID2 = 0.522600
VARI[  5] = "F EVID1 = 0.000000 EVID2 = 0.846869

RULES STRENGTHS
I = 0 CFRS1 = 0.360000 CFRS2 = 0.280800
I = 1 CFRS1 = 0.670000 CFRS2 = 0.522600
I = 2 CFRS1 = 0.870000 CFRS2 = 0.454662

I = 3 CFRS1 = 0.780000 CFRS2 = 0.780000

GOOD BYE
```

Example 2: **Untrained**

```
Give the file name which consists of Rule base one.dat

INPUT TO THE PROGRAM
No. of rules in the given file
rule [  0] is 0.9 E -> A
rule [  1] is 0.89 E -> A"
rule [  2] is 0.78 X -> Y
```

```
No. of evidences: 2

evidences E = 0.900000
evidences X = 0.560000

RULE STRENGHTS
CRFS[  0] = 0.900000
CRFS[  1] = 0.890000
CRFS[  2] = 0.780000

PREMISE STRENGTHS
VARI = A EVIDI = 0.000000 EVID2 = 0.00C000
VARI = E EVIDI = 0.670000 EVID2 = 0.00C000
VARI ="A EVIDI = 0.000000 EVID2 = 0.00C000
VARI = Y EVIDI = 0.000000 EVID2 = 0.00C000
VARI = X EVIDI = 0.560000 EVID2 = 0.00C000

NEURAL NET DEVELOPED
ONE  O E
SN  O A
ONE  1 E
SN  1 "A
ONE  2 X
SN  2 Y

After PROPAGATION values are

RULE strengths
CRFS1 = 0.900000 CFRS2 = 0.603000
CFRS1 = 0.890000 CFRS2 = 0.596300
CFRS1 = 0.780000 CFRS2 = 0.436800

PREMISE strengths
VAR = A  EVIDI = 0.000000 EVID2 = 0.243431
VAR = E  EVIDI • 0.670000 EVID2 = 0.670000
VAR = A~ EVIDI = 0.000000 EVID2 = 0.756569
VAR = Y  EVIDI = 0.000000 EVID2 - 0.436800
VAR - X  EVIDI = 0.560000 EVID2 = 0.560000

GIVE THE FILENAME WHICH CONSISTS OF DESIRED EVIDENCES

file name = one.evid

Reading desired evidences NS = 2
CAR[  0] = Y NEW[  0] = 0.390000
CAR[  1] = A NEW[  1] = 0.900000

SYSTEM IS NOT ABLE TO TRAIN THE NEURAL NET TO GIVE THE DESIRED
RESULT

INCONSISTENCY !!!

Inconsistency in Rule Base
Conflicting rules may be existing in the given rule base
Please check it
GOOD LUCK FOR THE NEXT TIME !!!

FINAL MODIFIED VALUES ARE
VARI[  0] = A  EVIDI = 0.748111 EVID2 = 0.099000
VARI[  1] = E  EVIDI = 1.000000 EVID2 = 1.000000
VARI[  2] = "A EVIDI = 0.000000 EVID2 = 0.901000
VARI[  3] = Y  EVIDI = 0.804688 EVID2 = 0.890000
VARI[  4] = X  EVIDI = 0.560000 EVID2 = 0.560000

RULE STRENGTHS
I = 0 CFRS1 = 0.603000 CFRS2 = 0.603000
I = 1 CFRS1 = 0.890000 CFRS2 = 0.890000
I = 2 CFRS1 = 0.780000 CFRS2 = 0.436800

GOOD BYE
```

6.2 Knowledge Representation

This section describes the process of Knowledge Representation in Neuro-Expert.

Representing knowledge in a computer consists of setting up a correspondence between a symbolic reasoning system and the outside world. This knowledge can be studied and understood in what we may call human terms, because the symbols used for its representation are seldom numerical. For example a Shares Advisor may use the following rule.

If the demand is good the company's share is good.

Such a rule would be given in the program's knowledge base in the following form.

```
(#rulel:

        (demand = good (0.67))        →
        (company = good (0.87))
```

The knowledge representation is done through a procedural language which is called as a Knowledge Definition Language and it is constructed according to a set of rules. This set of rules constitute the Grammar of the language.

6.2.1 Knowledge Definition Language (KDL)

The Knowledge Definition Language is the knowledge representation language used in Shell. This is a user friendly Logic programming language. The structure of the language is modular in nature. There can be any number of modules as long as there is enough memory with the systen.

A limited number of built-in-functions are provided as a part of the language.

The following part of this chapter describes the KDL in detail.

(1) Character set: KDL uses the letters A to Z (both upper and lower case), the digits 0 to 9, and certain special symbols as building blocks to form basic program elements (numbers, identifiers, expressions etc.)

The special symbols are listed below:

+	:	<=	]
-	;	>	{
*	,	>=	}
/	"	!=	#
:-	.	(	@
=	~	)	!
<	[	&	¦

(2) Identifiers: An identifier is a name that is given to some program element, such as variables, modules or main nodule. Identifiers are comprised of letters and digits, in any order, except that the first character must be a letter. Both upper and lower case are permitted and are considered to be indistinguishable. Under score can be used between any two characters as

connector. The maximum length of an identifier is 12 characters including connectors if any.

(3) **Numbers:** Numbers can be written in several different ways in KDL. In particular, a number can include a sign, **a** decimal point. Scientific notation is also allowed.

The following rules apply to all numbers.

(1) Commas and blank spaces cannot be included within the number.

(2) The number can be preceded by a plus sign (+) or a minus (-) sign if desired. If a sign does not appear, the number will be assumed to be positive.

(3) Numbers cannot exceed a specified maximum and minimum values. The range is 1.7E-308 to 1.7E+308.

(4) **Strings:** A String is a sequence of characters (i.e., letters, digits and special characters) enclosed by double quotes ("). Both upper and lower case can be used. The maximum number of characters that can be included in a string is 255 which is adequate for most purposes. Within a string only single quotes (') are allowed.

(5) **Data types:** One of the most important and interesting characters of KDL is its ability to support two data types. They are simple data type and compound data type.

Simple data type are single items that are associated with single identifiers on a one-to-one basis. There are three single data types. They are numeric, data, and boolean. An identifier with a data type numeric can hold a real number of any form. An identifier with a data type can hold a valid date of ten character length.

The identifier with a boolean data type can hold a TRUE of FALSE value. Boolean type data are truth values that are either true or false.

Compound data type consist of multiple items that are related to one another in some specific manner. Each group of items is associated with one identifier. The KDL supports string data type. The identifier with a string data type can hold a string of 255 character length.

(6) Constants: The KDL has two built in constants, viz. TRUE and FALSE. These hold a True/false value or Yes/Mo value. These constants can be used in the statements to initialize a boolean type identifier. The user will not be allowed to redefine these constants.

(7) Variables: An identifier whose value is allowed to change during the execution of the knowledge is called a variable. The data type of the variable will be automatically determined by interpreter depending on the first usage of the variable. Later it will not be allowed to use as another data type.

For example, if a variable say DATE is used in a date function, then variable DATE will be considered as a date type data. It cannot be referred to as numeric, boolean or string data type later in the knowledge base.

The data type of database fields will be kept as such through a consultation. Data type conflicts between user defined variable and database fields are not allowed.

(8) **Rule number:** The rule number is label which identifies each rule. The rule number, comprises of letter, digits and hyphen. Both upper and lower case letters are allowed. The rule number should start with character hash (#). The ending character cannot **be** a hyphen and two or more consecutive hyphens are not allowed. The maximum length of a rule number is 12 character including the hyphens. A rule number should be unique throughout a knowledge base.

eg: #23-03-A, #ABCDE-001

(9) Statements **and** assignments: The KDL statement can be a function, arithmetic statement or a group of rules. There are two basic types of statements in KDL, viz., simple and compound statements. The simple statements are essentially single, unconditional instructions that perform one of the following tasks.

(1) Assign a data item to a variable or assign an expression to a variable. This is called an assignment statement.

(2) Access to a system function.

(3) Access to a module.

Some typical examples for assignment statements
ASCST:- TRUE
GPAY:- BASIC + DA + HRA;

(10) Functions/commands: KDL contains a number of standard functions that are used with various data types. These functions can also be called as built-in functions. All the functions return TRUE on success and FALSE on failure. Some of these functions accept parameters.

Variable must be assigned, some value using these functions, before using them **within** the RULES statement.

Example: 1 REFERDB "mydata.dbf", EMP_NO; This function refers to the database file MYDATA.DBF and loads the record using the value of key field EMP_NO obtained from the user.

Example: 2 ASK "Are you a permanent employee" to PERMANENT;

The ASK function prompts the user and accepts the TRUE or FALSE value and stores in the variable PERMANENT.

(11) Rules: Basically the knowledge base is constituted of different modules. All the modules have the same status except main module. When the knowledge base is executed, the main nodule guides the inference engine through the other modules.

A module is constituted of different statements which can be arithmetic expressions, function calls (no user defined functions are allowed) and rules.

To every legal statement, called a premise of proposition, one of the two possible values TRUE and FALSE is assigned; these are often called Boolean Values, after the mathematician and logician George Boole (1815-1864). Complex propositions can be expressed by using logical connectives written as follows.

AND	&
OR	¦
NOT	!

The Inference Engine expects either TRUE or a FALSE value from a statement and if the returned value is TRUE execution continues to the next statement else execution of the knowledge base is stopped.

Rules of nodes in the knowledge base are represented as follows:

#<rule number>:

i) Rule number is an alphanumeric string followed by a colon (:). The allowed separator is hyphen (-). Rule number should not start or end with the separator.

   Rule number length should not exceed 11 characters including the separator. Spaces are also not allowed inside a rule number. Rule numbers should be unique in any Knowledge Base.

ii) Conditions are expressions which evaluate to a boolean value. Each condition should be enclosed within brackets and must specify the premise strength in roundbrackets.
   e.g. (demand = OK (0.5))

The comparators comprise of the following:

<	less than
>	greater than
=	equal to
<=	less than or equal to
>=	greater than or equal to
!=	not equal to

Variables can be numeric, string or boolean types depending on the function used to obtain the value. The variables should have a value before it is being used in a c.f. explanation range (0 to 1.0) etc.

The knowledge representation language is designed in such a way to avoid the complexities of the conventional languages in developing expert system applications. The structure of the Knowledge definition language is more like that of a Fourth Generation Language (4GL). This gives the users more flexibility and easy means to represent their logic.

iii) Conclusions: The conclusions are specified after the reserved "→" symbol and are specified in round brackets.

e.g. (advice = "Buy the share)

iv) Rule Strength: The rule strength must be specified at the end of rule specification enclosed in round brackets. The rule strength would be a float value in the range -1.0 and +1.0.

e.g.

```
(#rulel:
(demand = ok (0.78))

(advice = "Buy the share ") (0.9))
```

6.2.2 Built-in Functions

The following are the built-in-functions provided in the Knowledge Definition Language. Each of the function returns a TRUE or FALSE value to the Inference Engine depending on success or failure. A failure will force the Inference Engine to stop execution. No user defined functions are allowed except that external executable programs can be executed with the built-in-functions CALL.

*ASK	*CALL	*DATEDIFF	*GETDATE
*GETNUM	*GETSTRING	*GETSYSDATE	*WRITE
*MODULE	*RUN	*REFERDB	*LOAD

A brief discussion on the above functions follows:

(1) ASK

USAGE : ASK <"prompt"> to <var name>

PARAMETERS

"prompt" :	The literal text (enclosed in quotes) can be displayed by the ASK clause.
Variable :	Any valid variable name. If the variable is a new one, then it will be considered as a BOOLEAN type.

DESCRIPTION	The ASK statement displays its prompt message to the user, then waits for a response which is in the form of YES OR NO. The value entered by the user is assigned to the variable.

(2) CALL

USAGE	CALL <"filename">
PARAMETERS	
filename	The executable program name within double quotes.
DESCRIPTION	The CALL clause executes a DOS executable file (viz. files with .com, .exe, .bat extensions).

(3) DATEDIFF

USAGE	DATEDIFF <DATEDIFF <Datel>, <Date2> to <Variable>
PARAMETERS	
Date1	The variable should be of type Date and it should contain a valid Date.
Date2	The variable should be of type Date and it should contain a valid Date.
Variable	The difference between the dates is assigned to the variable and it should be a numeric type.

DESCRIPTION	Finds the difference between the two given dates and stores the number of days in the given numeric variable.

(4) GETDATE

USAGE	GETDATE <"prompt"> to <variable>
PARAMETERS	
"prompt"	The prompt string (enclosed in quotes) to be displayed.
Variable	If the variable is new one, then it will be considered as a Date type. Otherwise it should be DATE type.
DESCRIPTION	The GETDATE statement displays its message to the user, then accepts the date and assigns to the variable. If the user enters an invalid date then an error indicating beep sour.d is produced and the system prompt to enter a valid date.

(5) GETSYSDATE

USAGE	GETSYSDATE to <date>
PARAMETERS	
date	The system date is assigned to the variable <date>

DESCRIPTION	The system date is to be set before using this function, if you do not have real time clock in the system. This function gets the date from the system and is stored in date.

(6) GETNUM

USAGE	GETNUM <"prompt"> to <variable>
PARAMETERS	
"prompt"	The prompt string (enclosed in double quotes) to be displayed.
variable	Any valid variable name. If the variable is a new one, then it will be considered as a numeric type. Otherwise it should be a numeric type.
DESCRIPTION	The GETNUM statement displays its prompt to the user, then waits for a response to accept the number to be supplied by the user. The actual number entered by the user is assigned to the variable.

(7) GETSTRING

USAGE	GETSTRING <"prompt"> to <variable>
PARAMETERS	
"prompt"	The prompt string (enclosed in double quotes) to be displayed.

variable	Any valid string type. When defined for the first time, the variable will be considered to be string type.
DESCRIPTION	The GETSTRING statement display the message to the user, then waits for a response to accept the string to be supplied by the user. The actual string entered by the user is assigned to the variable.

(8) WRITE

USAGE	WRITE <" user prompt" > [<variable>,<endln>]

PARMETERS

User prompt	Any valid message string.
Variable	Any valid variable
Endln	Newline
DESCRIPTION	This inbuilt function allows the display of user messages and the prompts on the screen. This is the output statement in KDL and also has provisions for new lines if specified.

(9) MODULE

USAGE	MODULE <module name>

PARMETERS	
Module name	The name of the module to be loaded or called
DESCRIPTION	This in-built function allows the knowledge engineer to split up the rules into modules in order to support modular and structured programming. Each module could be called using this function.

(10) RUN

USAGE	RUN
PARMETERS	None
DESCRIPTION	This in-built function allows the user to run the currently loaded knowledge base.

(11) LOAD

USAGE	LOAD <knowledge base>
PARMETERS	Knowledge Base : The name of the knowledge base to be loaded.
DESCRIPTION	This command loads the specified knowledge base. The system assumes that each knowledge base would have ".knb" as the extension.

(12) REFERDB

USAGE	REFERDB <"filename">, <key field>

PARAMETERS

Filename	:	The Database file name within double quotes
Key field	:	key field of the database record.
DESCRIPTION	:	The REFERDB clause stores the first record matching the key field after getting the key field value from the user.

6.2.3 Sample Knowledge Base

This sub-section presents a window into the shares knowledge base. The actual knowledge base comprises of more than 200 rules, however, we present here only some sar.ple rules.

Rules to grade price

```
(#ruleprl:

   (max_p -a  (0.67)) &
   (price = a  (0.78))
→
   (p_grade = a)  (0.9))

(#rulepr2:

   (max_p = b  (0.8))  &
   (avg_p = c  (0.78))  &
```

```
    (price < b  (1.00)) &
    (price >= c (1.00))
→
    (p_grade = b) (0.9))

(#rulepr3

    (avg_p = a (0.89)) &
    (min_p = b (0.78)) &
    (price  < a (0.89)) &    (price >= b (0.89))

→
    (p_grade =  c) (0.93))
```

Rules to grade transport cost

```
(#ruletcl:
    (max_t = a (0.89))
    (t_cost = a (0.9))
→
    (t_grade =  a) (0.94))
(#ruletc2:
    (max_t = a (0.89))
    (avg_t = b (0.89))
    (t_cost  < a (0.9))    (t_cost  >= b (0.9))
→
    (t_grade =  b) (0.94))

(#ruletc3:

    (avg_t = a (0.89))
    (min_t = b (0.89))
    (t_cost < a (0.9))
   (t_cost >=b (0.94))
→
    (t_grade =  c) (0.94))
```

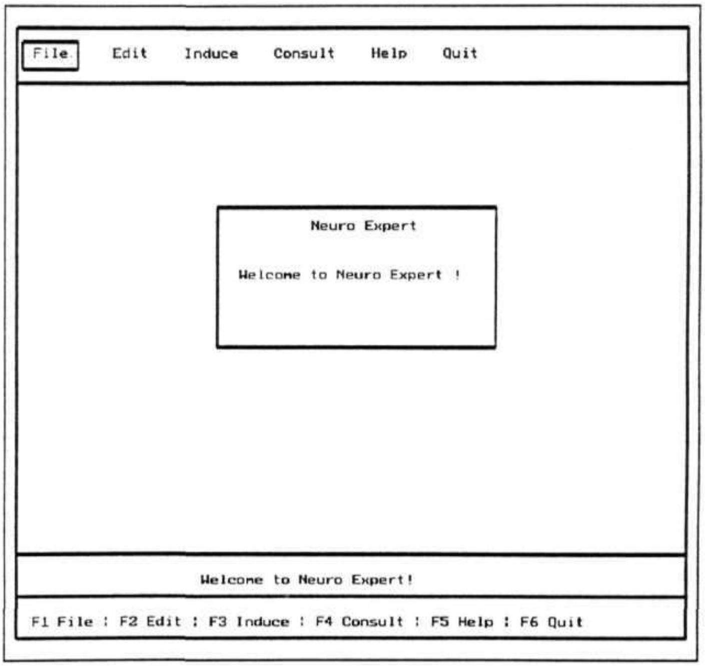
Rules to grade demand

```
(#ruledl:

    (p_grade = c (0.89))
    (t_grade = c (0.89))
    (brand = good (0.9))
→
    (write "demand is very high"]
    (demand = vhigh) (0.9))

(#ruled2:
    (p_grade = b (0.89))
    (t_grade = c (0.89))
    (brand = good (0.9))
→
    (write "demand is high")
    (demand = high) (0.94))

(#ruled3:

    (p_grade = a (0.89))
    (t_grade = c (0.89))
    (brand = good (.90))
 →
    (write "demand is ok")
    (demand ok) (0.98))
```

6.3 User Interface and Consult Facility

The User Interface in Neuro Expert is built-up **of using the** basic screen handling routines. It provides basic navigations with the help of pull-down menu structure, Context-sensitive help, Integrated Editor and basic DOS utilities.

The main menu of neuro expert: The Main Menu of Neuro Expert consists of six options. The options can also be selected



Neuro Expert

Welcome to Neuro Expert !

Welcome to Neuro Expert!

F1 File ¦ F2 Edit ¦ F3 Induce ¦ F4 Consult ¦ F5 Help ¦ F6 Quit

either by selecting the required option from the ring menu or by pressing the appropriate hot keys as displayed at the status bar at the bottom of the screen. The selected option will be highlighted and on selection **the corresponding help message** would be displayed in the message box.

The available options are

F1 File
F2 Edit
F3 Induce
F4 Consult
F6 Help
F10 Quit

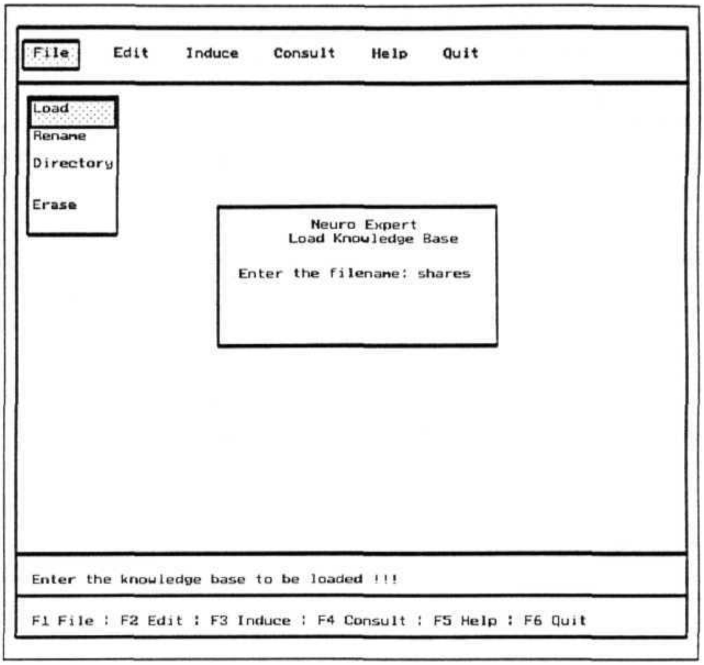
Let us now have a detailed look at all the options.

6.3.1 File

The files menu allows to access DOS file commands without quitting Neuro Expert. This option displays a pop-up menu which consists of the following choices:

(a) Load
(b) Rename
(c) Directory
(d) Erase

These can be selected by pressing Fl key from the main menu. The choices can be selected using arrow keys and pressing return. The menu options can also be selected pressing the upper case letter which is displayed in each menu option. If an undefined key is pressed, the system produces a beep sound.



Neuro Expert
Load Knowledge Base

Enter the filename: shares

Enter the knowledge base to be loaded !!!

F1 File : F2 Edit : F3 Induce : F4 Consult : F5 Help : F6 Quit

Each option of the menu is described below.

6.3.1.1 Load

Knowledge base should be loaded into the memory before consult or edit is invoked. Using the load option, the contents of the knowledge base file can be transferred to the system memory.

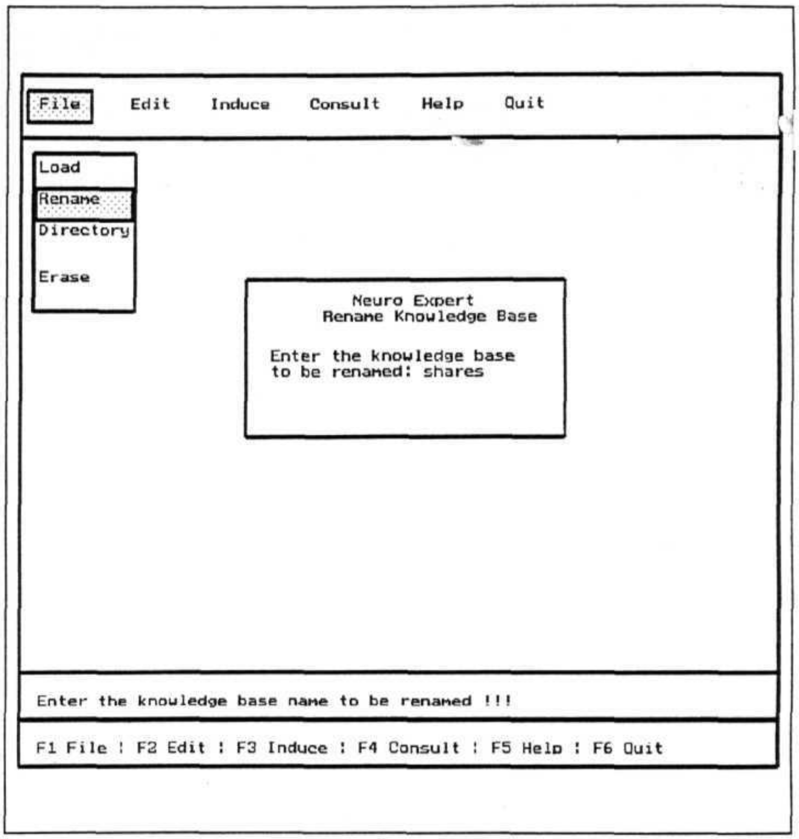
Extension for the knowledge base file name should be .KNB. Other file name extensions are not valid for the knowledge base file.

All cursor keys can be used to edit the path name. Left and right arrow keys can be used to move through the path name. The Home key takes the cursor to the beginning and the End key to the end of the path name. Del key erases the character in the cursor position and Backspace key erases the character in the left side cursor. Editing can be aborted by pressing Esc key and the system retains the old file loaded. After editing the filename Enter key should be pressed to load the file.

If the system finds errors in the path of file name or if the file is not found, it displays error messages. If the file is successfully loaded, the file name and the number of lines loaded are displayed in the title window at the top of the screen.

6.3.1.2 Rename

This option lets the user to rename a file. If it is only required to rename the file, enter a new name. The Rename option when used to rename a file.



Neuro Expert
Rename Knowledge Base

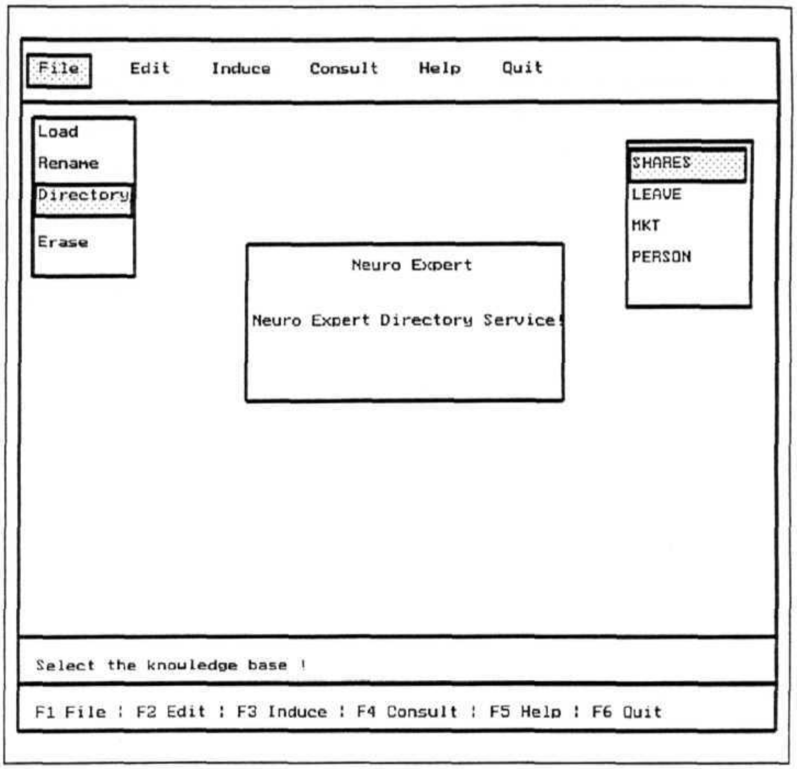
Enter the knowledge base
to be renamed: shares

Enter the knowledge base name to be renamed !!!

F1 File ¦ F2 Edit ¦ F3 Induce ¦ F4 Consult ¦ F5 Help ¦ F6 Quit

6.3.1.3 Directory

The directory option displays the list of **knowledge bases in** current working directory as with the DOS command "dir *.knb"



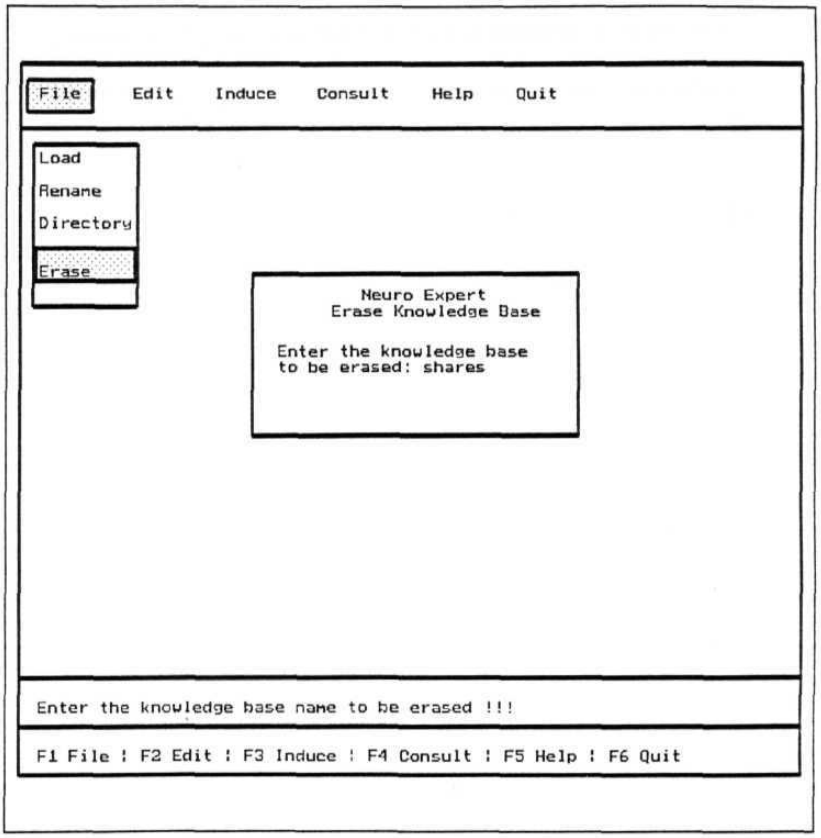
Neuro Expert

Neuro Expert Directory Service!

Select the knowledge base !

6.3.1.4 Erase

Choosing this option lets the user delete a file from the disk. When this option is selected Neuro Expert prompts for the name of the file to erase. After selecting the file to delete, Neuro Expert erases the file from the disk. This command is equivalent to DOS command delete.



Neuro Expert
Erase Knowledge Base

Enter the knowledge base
to be erased: shares

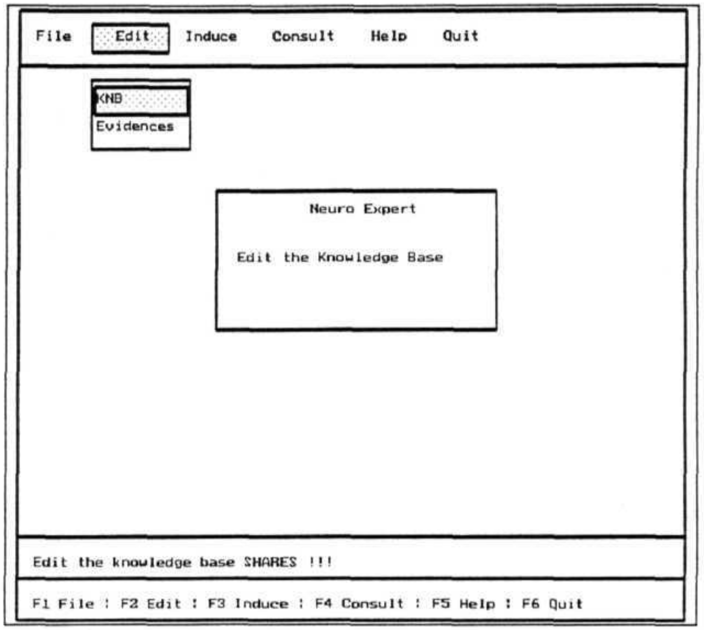
Enter the knowledge base name to be erased !!!

F1 File ¦ F2 Edit ¦ F3 Induce ¦ F4 Consult ¦ F5 Help ¦ F6 Quit

6.3.2 Edit

The edit option allows the user to edit the current knowledge base and the desired evidences. The edit option further displays a sub-menu with two sub-options:

- Knowledge Base
- Desired Evidences


File
Edit
Induce
Consult
Help
Quit
KNB
Evidences

Neuro Expert

Edit the Knowledge Base

Edit the knowledge base SHARES !!!

F1 File : F2 Edit : F3 Induce : F4 Consult : F5 Help : F6 Quit

6.3.2.1 KNB

This sub-option allows the user to edit the current knowledge base. The option automatically opens the corresponding ".knb" file using the integrated system editor, "NE". The user can modify the rules using the normal editing keys and save the changes and come back to the main menu.

6.3.2.2 Desired evidences

This sub-option allows the user to edit **the** desired **evidence** file for the current knowledge base. The option automatically opens the corresponding ".evi" file using the integrated system editor, "NE". The user can modify the rules using the normal editing keys and save the changes and come back to the main menu.

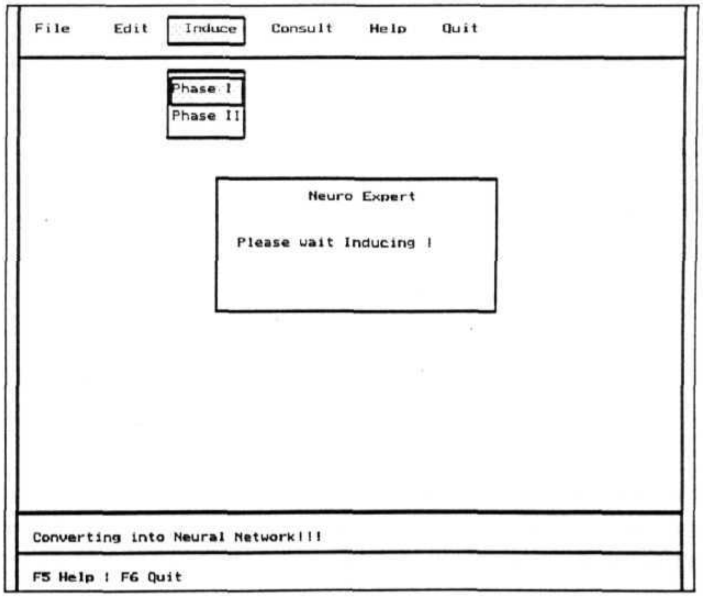
6.3.3 Induce

This option would allow the user to derive appropriate CFs for the rules in the knowledge base and also indicate whether the knowledge base is consistent or not. Internally this option comprises of two phases:

- Mapping Rules to ANN (Phase I)
- Training the ANN (Phase II)

6.3.3.1 Phase I

During the first phase the rules of the currently loaded knowldege base are mapped to Nodes of ANN.



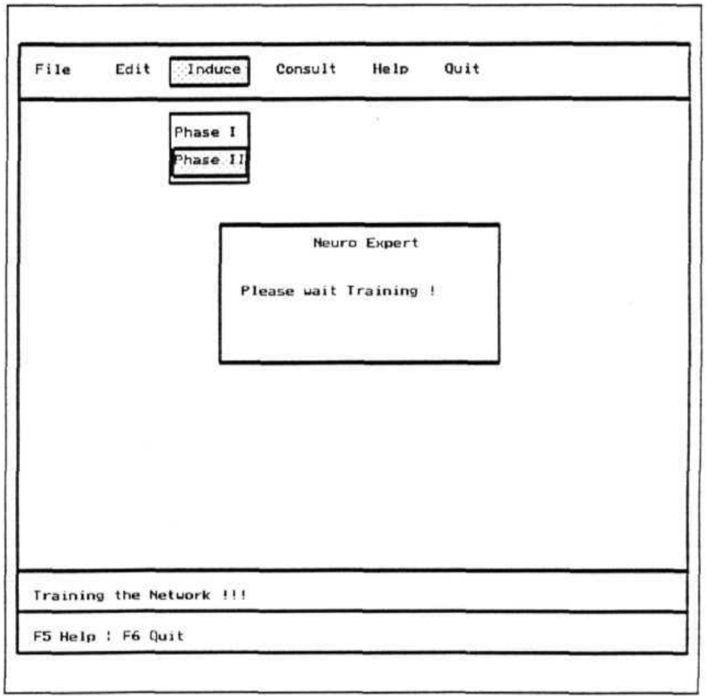
Neuro Expert

Please wait Inducing !

Converting into Neural Network!!!

6.3.3.2 Phase II

During the second phase, the ANN is **trained** to achieve the desired evidences as specified in ".evi" file by the user. If the training succeeds then the current knowledge base **is** updated with the new premise strengths and evidences.



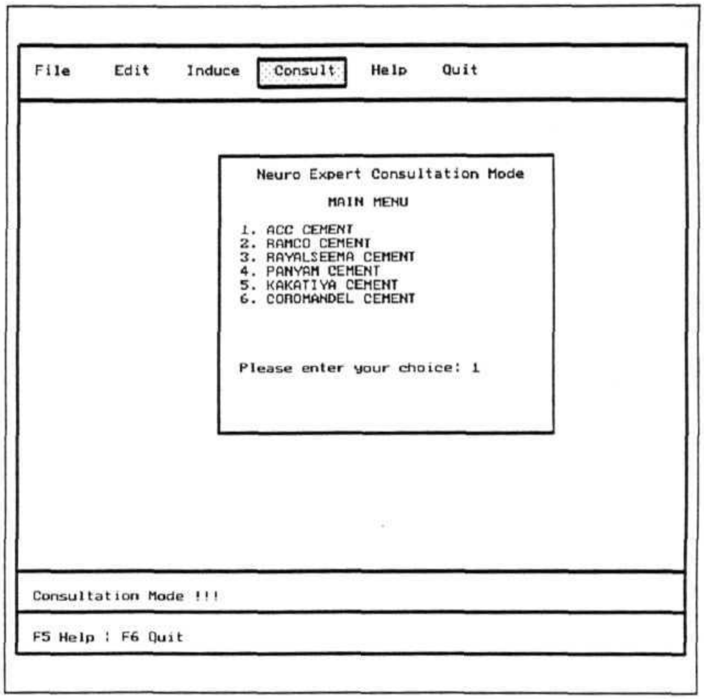
Please wait Training !

Training the Network !!!

F5 Help : F6 Quit

6.3.4 Consult

The Consult option should be selected after loading **the** knowledge base. The knowledge base can be loaded through **the** files menu. Consult option initiates the interactive **session** with the users, wherein the user gives his specifications and based on the user inputs and the trained knowledge base the inference engine fires the corresponding rules and arrives at a conclusion.

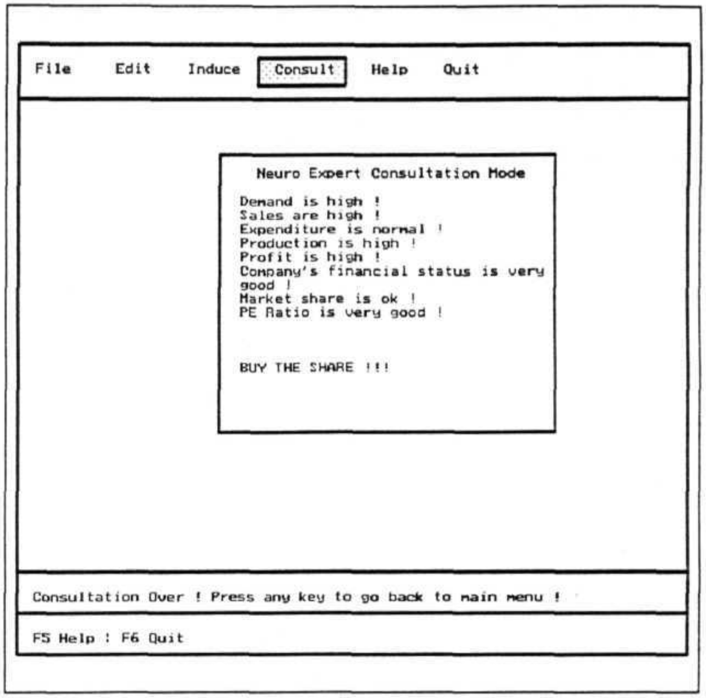


```
Neuro Expert Consultation Mode

        MAIN MENU

1. ACC CEMENT
2. RAMCO CEMENT
3. RAYALSEEMA CEMENT
4. PANYAM CEMENT
5. KAKATIYA CEMENT
6. COROMANDEL CEMENT


Please enter your choice: 1
```

Consultation Mode !!!

F5 Help : F6 Quit

Depending on the user input i.e. the company selected in this case Neuro Expert would provide advice and also the line of reasoning behind it. For instance if the user selects ACC cement as the company name then the following advice would be flashed on the screen.



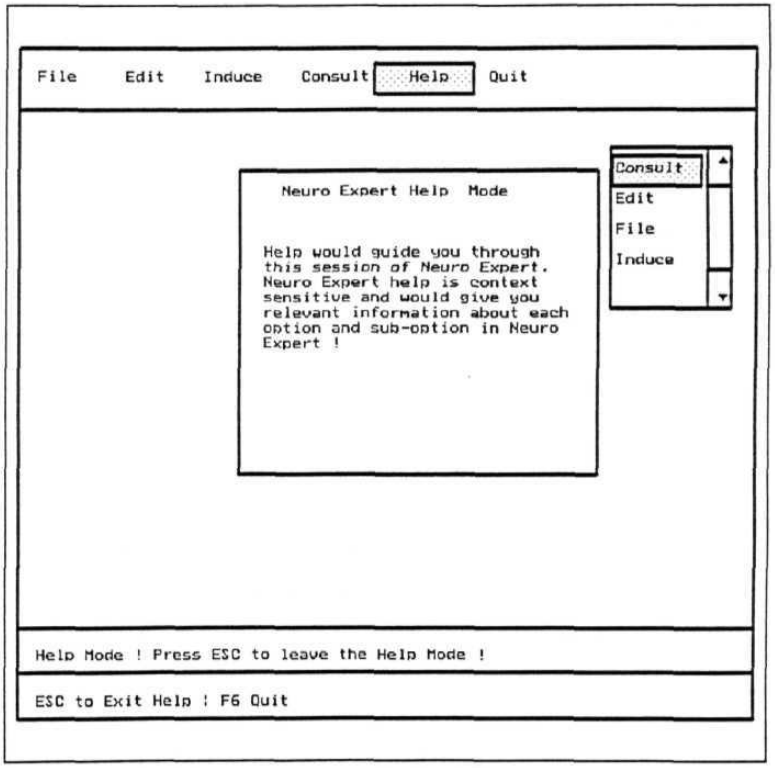
```
Neuro Expert Consultation Mode

Demand is high !
Sales are high !
Expenditure is normal !
Production is high !
Profit is high !
Company's financial status is very
good !
Market share is ok !
PE Ratio is very good !


BUY THE SHARE !!!
```

```
Consultation Over ! Press any key to go back to main menu !
```

6.3.5 **Help**

The Neuro Expert help is a context sensitive help which gives advice pertaining to the topic from **where** help is requested. The help is also provided **with** a table of help indices. Using arrow keys and pressing return, a topic can be selected. HOME key and END key to be incorporated in barmenu function. PGUP key brings the selection bar to the top of the current column in the index window and PGDN key to the bottom of the current column in the index window.

The help can be invoked at any instant by pressing F5 key. If F5 key is pressed while in the help session the index for the help will be made available for selecting another topic. The arrows at the top and bottom right corners of the help screen indicate whether more information is available in the preceding or proceeding page. PGDN/PGUP or up arrow, down arrow keys can be used to scan through the topic selected.



Neuro Expert Help Mode

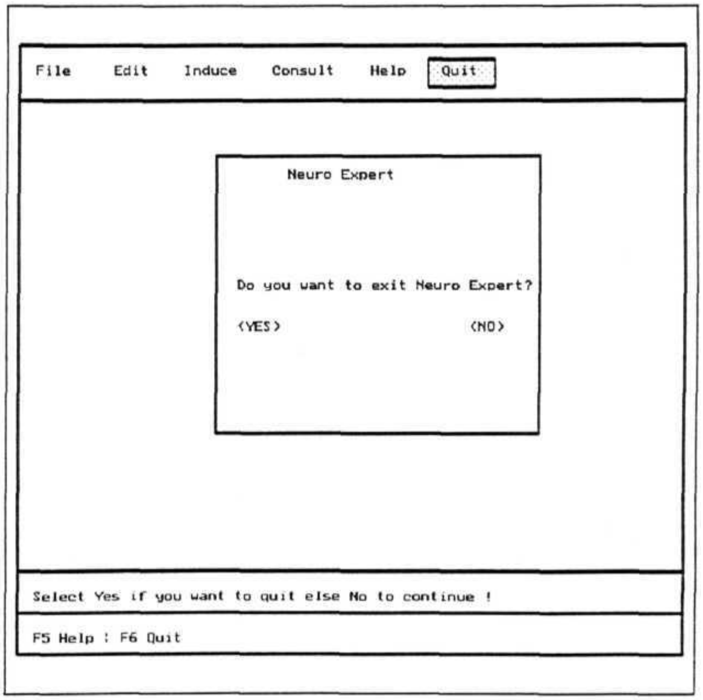
Help would guide you through this session of Neuro Expert. Neuro Expert help is context sensitive and would give you relevant information about each option and sub-option in Neuro Expert !

Help Mode ! Press ESC to leave the Help Mode !

ESC to Exit Help ¦ F6 Quit

6.3.6 Quit

The user can quit Neuro Expert by either selecting quit option from the main menu or by pressing F6. The system displays a confirmation box and confirms the selection. At any instant the user can quit from any pop up menu without selecting an option by pressing Esc key.

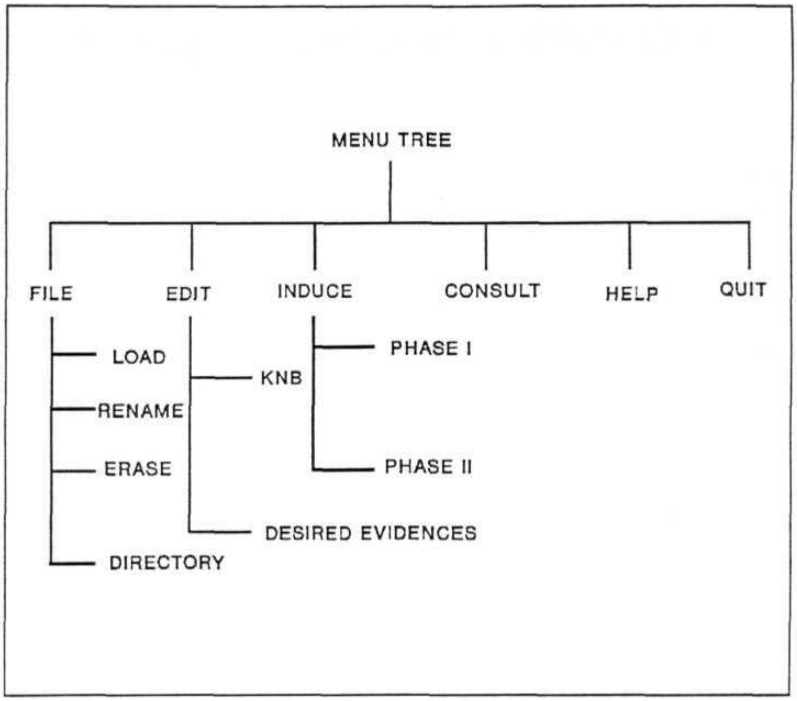


Do you want to exit Neuro Expert?

Select Yes if you want to quit else No to continue !

6.3.7 Neuro Expert Menu Command Tree

A graphical representation of the menus provided with Neuro Expert is shown in the following figure.



MENU TREE
FILE
EDIT
INDUCE
CONSULT
HELP
QUIT
LOAD
RENAME
ERASE
DIRECTORY
KNB
DESIRED EVIDENCES
PHASE I
PHASE II

Chapter 7

Summary and Future Directions

The modern world with its complexities and super-specializations, has reached a stage where expertise is scarce. In most fields there are more problems than experts. More often there are tines when access to knowledge, experience and judgement of an expert in the field become invaluable assets and are essential for timely decision making. Knowledge itself is about to become the new wealth of nations.

Significant efforts have been directed towards AI, to emulate the reasoning process of human experts. Expert System (sub-domain of AI) attempts to reduce the scarcity of human experts when the modern world finds itself in the midst of complexities, super-specializations and cut-throat competition. Expert systems act as intelligent assistants to human experts and contribute a great deal to business organization and productivity. An expert system should be able to explain the reasoning process that lead to its conclusions. Most of the expert systems developed so far have production rules as their knowledge representation schemes and are termed as rule based expert systems. Rule-based expert systems are referred to as conventional expert systems (CES). The CES can solve a wide range of complex problems by selecting relevant rules and combining their results in appropriate ways.

We have examined the requirements of intelligent systems to meet the demands of Management Applications **and** their necessary performance requirements. These are:

(a) To respond to situations very flexibly.
(b) To make sense out of ambiguous or contradictory messages.
(c) To recognize the relative importance of different elements of a situation.
(d) To find dissimilarities between situations despite similarities which may link them.

The rigid framework of conventional expert systems has made expert systems development for management applications a gruelling process. Though CES can solve a variety of problems certain problems are encountered specially in management applications where decisions have to be made very fast and the output from the system needs to be as reliable as the experts output. A typical application is expert systems for the share market.

The specific problems of CES are:

1. Rules can interact only through working memory. This is a major bottleneck if there are numerous rules to be processed.

2. In a situation where multiple hypotheses are to be dealt with, forward chaining method is applied. In such cases the CES can rarely explore all the alternatives from among the available multiple hypotheses. It is desirable that search be continued to explore all alternatives so that these can be ranked and the best possible hypothesis be selected.

3. The refinement of a rule base is a difficult problem. Refinement is done by: interaction with human experts, machine learning, justification or explanation based on domain theory, or empirical refinement.

These approaches are expensive and involve a lot of time and effort. Minimizing the involvement of experts again and again can substantially reduce the cost of maintaining expert systems.

4. A Conventional Expert System primarily conducts symbolic reasoning but uncertainty is often handled numerically. The CES using uncertainty paradigm does not have systematic methodology for debugging a given factual information to assert a conclusion with desired certainty factor given by an expert.

In the present work a system that can overcome many of the drawbacks mentioned above is presented. This system enhances the applied epistemics and is ideal for developing expert systems on management applications. The system aims at achieving high reliability through major architectural innovations that avoid the traditional bottlenecks. Knowledge-based neural network (KBNN) is integrated into CES. This combination results in a high reliability level and high throughput that are essential for management applications.

It is vital that the expert system gives outputs that are close to the opinion of an expert. That is the certainty factor given by the Expert and the system should be as close as possible. To achieve this kind of accuracy a special Neural Network (NN) has been developed on modified back-propagation methodology.

A framework has been developed whereby the rules from the rule base can be mapped onto this specially developed network. A very significant feature of our system is that a methodology has been developed that can utilize non-binary inputs. So far this has never been achieved.

Each domain attribute or concept is mapped into a neural node and each premise is mapped into a connection. Knowledge embedded in such networks accounts for their faster convergence to a desired stage in the learning phase. The knowledge of a NN lies in its connections and associated weights.

In conventional expert system rules can interact only through the working memory, this presents a problem when numerous rules are to be processed. By mapping the rule base into a neural net several rules are fired when their premises are activated. Thus the rule interaction becomes distributed over the network rather than centralized through the working memory. This significantly improves the system performance.

To conclude, the most important advantage of using NN, is that the NN can be trained to reach a level closest to the expert's opinion. This system is capable of asserting a conclusion with the desired Certainty factor (CF). This adds dynamism to expert system (accommodate changes).

As an example the share market is taken. An analysis of equity investment is presented. Both fundamental analysis and technical analysis are discussed. Fundamental analysis is a value based approach. Technical analysis is a market based approach.

Fundamental analysis comprises of three phases

1. Economic Analysis
2. Industry Analysis
3. Company Analysis

Technical analysis emphasizes on prices, price changes and trading volumes.

As a sample, the Cement industry has been taken and rules formulated.

A rule base has been constructed and these rules have been mapped into Neural Network, for which we have developed a Neuro-Expert system.

Apart from the usual features such as lexical analyzer, interpreter, knowledge base, user interface, inference engine, etc. , this system has an improved and more powerful knowledge definition language (KDL), for knowledge representation.

This is a user-friendly logic programming language. the structure of this language is modular in nature. There is a possibility of having any number of modules depending upon the memory of the system. Its structure is more like that of a fourth generation language (4GL). This gives the knowledge engineer more flexibility and he can represent his logic with ease.

For the convenience of the user a database interface is integrated with the system using Foxpro. This facilitates the job of the user and he can interact with the system without having any technical knowledge. He can even form rules and enter rules through this utility into the system. The rules can be directly entered into the knowledge base through this utility.

This system is capable of mapping the rules to NN through the built-in utility and the NN is trained according to the required CFs.

Thus our system enhances the applied epistemics of an Expert System and can be used by any management professional who is not a computer expert.

Separate trials were conducted with satisfactory results. However, we felt that there were certain areas where there was scope for further work.

Future Extensions

1. We have not presented a detailed methodology for pinpointing inconsistencies of rules. We have just provided a detection mechanism that detects inconsistency while training the NN. A detailed methodology for pin-pointing inconsistencies can be developed.

2. To study adaption of dynamic neural networks, whereby nodes can be deleted and added dynamically depending on the network capabilities in dealing with the task.

3. To modify the network, to carry out predictions related with equity shares, based on past statistics.

4. A more friendly GUI is desirable.

5. The system viability is to be assessed for other management decision-making areas such as personality assessment, performance assessment etc.

6. Explore alternative network paradigms for management applications.

References

Aggarwal, V.S., 1985. Share Capital and Debentures. New Delhi, India: National Publishing House.

Aleksander.I., and Morton. H., 1990. An Introduction to Neural Computing. London, UK: Chapman and Hall.

Aoki, M. 1967. Optimization of stochastic systems—Topics in discrete-time systems. New York: Academic Press.

Barto, A.G., Sutton, R.S., and Anderson, C.W. 1983. Neuronlike adaptive elements that can solve difficult learning control problems. Institute of Electrical and Electronics Engineers Transactions, SMC-13. 834-846 pp.

Barr, A., and Feigenbaum, E.A. (eds). 1982. The Handbook of Artificial Intelligence. Vol 1-3. Los Altos, California, USA: William Kaufmann.

Baum, E.B., Moody, J., and Wilczek, F. 1986. Internal representations for associative memory. NSF-ITP-86-138 Institute for Theoretical Physics. Santa Barbara, California: University of California.

Baxt, W.G. 1992. Improving the Accuracy of an Artificial Neural Network using Mutiple Differently trained Networks. Journal of Neural Computation, 4, 772-780 pp.

Bennett, J.S., and Englemore, R.S. 1979. SACON: A knowledge-based consultant for structural analysis. International Joint Conferences on Artificial Intelligence 6.

Bottou, L., and Vapnik, V. 1992. Local Learning Algorithms. Neural Computation, 4, 888-900 pp.

Buchanan, B.G., and Shortliffe, E.H., (eds) 1984. Rule-based expert systems. Reading MA USA: Addison-Wesley.

Burr. 1987. Experiments with a connectionist text reader. Pages 717-724 in Proceedings of the IEEE First International Conference on Neural Networks, Vol. 4., San Diego.

Caianiello, E.R. 1961. Outline of a theory of thought-processes and thinking machines. Journal of Theoretical Biology 2:204-235.

Cohen, M.A., and Grossberg, S. 1983. Absolute stability of global pattern formation and parallel memory storage by competitive neural networks. IEEE Trans. Syst. Man Cybern. SMC-13, 815-826 pp.

Cottrel, Munro, and Zipser. 1987. Image compression by back-propogation: An example of extensional programing, ICS Report 8702. San Diego: University of California.

Davis, R. 1976. Application of meta-level knowledge to the construction, maintenance, and use of large knowledge base. Ph.D. thesis. Computer Science Dept., Stanford University, USA.

Doyle, J. 1979. A truth maintenance system. Artif. Intell. Vol. 12, No. 3:231-272.

Duda, R.O., and Hart, P.E. **1973.** Pattern classification and scene analysis. New York: John Wiley & Sons.

Duda, R.O., Gaschnig, J.G., and Hart, P.E. 1979. Model design in the PROSPECTOR consultant system for mineral exploration. A chapter in Expert systems in the micro-electronic age (Michie, D., ed.). Edinburgh: Edinburgh University Press.

Engelmore, R.E., and Terry, A. 1979. Structure and function of the CRYSALIS system. International Joint Conference on Artificial Intelligence 6.

Eshelman, L., and McDermott, J. 1986. MOLE: A knowledge acquisition tool that uses its head. Pages 950-955 in Proc. AAAI-86, Philadelphia, PA.

Farley, B.G., and Clark, W.A. 1954. Simulation of self-organizing systems by digital computer. Institute of Radio Engineers—Transactions of Professional Group of Information Theory, PGIT-4. 76-84 pp.

Feldman, J.A., Fanty, M.A., Goddard, N.H., and Lynne, K.J. 1988. Computing with structured connectionist networks. Commun. ACM Vol. 31, No. 2:170-187.

Feldman, J.A., and Ballard, D.H. 1982. Connectionist models and their properties. Cognitive Sci. Vol 6, No. 3:205-254.

Frost, R. 1988. Introduction to Knowledge Based Systems.

Fu, L.M. 1985. Learning object-level and meta-level knowledge in expert systems. Ph.D. thesis. USA: Stanford University.

Fu, L.M. and Fu, L.C., 1990. Mapping Rule Based Systems into Neural Architecture. Knowledge Based Systems 3(1) (1990)48-56.

Fu , L., 1991. Knowledge based refinement by back-propagation. Data and Knowledge Engineering 7(1991) 35-46.

Gallager, R.G. 1968. Information theory and reliable communication. New York: John Wiley & Sons.

Gallant, B.I. 1988. Connectionist expert systems. **Commun. ACM** Vol. 31, No. 2:152-169.

Geffner, and Pearl. 1987. On the probabilistic semantics of connectionist networks. Pages 187-195 in Proc. **1st** IEEE Internat. Conf. and Neural Networks, San Diego, CA.

Ginsberg, A., Weiss, S.M., and Politakis, P. 1988. Automatic knowledge base refinement for classification systems. Artificial Intelligence 35(2):197-226.

Gold, B. 1986. Hopfield model applied to vowel and consonant discrimination. MIT Lincoln Laboratory Technical Report, TR-747, AD-A169742 (June 1986).

Grossberg, **S. 1986.** The adaptive brain I: Cognition, learning, reinforcement, and rhythm, and The adaptive brain II: Vision, speech, language, and motor control. Amsterdam: Elsevier/North-Holland.

Grossberg, S. 1982. Studies of mind and brain: Neural principles of learning, perception, development, cognition, and motor control. Amsterdam: Reidel Press.

Hayes-Roth, F., Waterman, D.A., and Lenat, D.B. (eds). 1983. Building Expert Systems. Reading MA, USA: Addison-Wesley.

Hinton, G.E., Sejnowski, T.J. and Ackley, D.H. 1984. Boltzmann machines: Constraint satisfaction networks that learn. (Tech. **Re.** No. CMU-CS-84-119). Pittsburgh, PA: Carnegie Mellon University.

Hinton, G., and Sejnowski, T. 1983. Optimal perceptual inference. Pages 448-453 in Proc. IEEE Computer Society Conf. on Computer Vision and Pattern Recognition.

Hoffmann, J., Skrzypek,J. and Vidal,J.J. 1993. Cluster Network for Recognition of Handwritten Cursive Script Characters. Network Networks, Vol 6(1) 69-78 pp.

Holsapple, C.W., and Whinston, A.B. 1989. Business expert systems. New Delhi, India: Galgotia Publications Pvt. Ltd.

Hopfield, J.J. 1984. Neurons with graded response have collective computational properties like those of two-state neurons. Pages 3088-3092 in Proc. Natl. Acad. Sci. USA, Vol. 81 (May 1984).

Hopfield, J.J. 1982. Neural networks and physical systems with emergent collective computational abilities. Pages 2554-2558 in Proc. Natl. Acad. Sci. USA, Vol. 79 (April 1982).

Jones, W.P., and Hoskins, J. 1987. Back-propogation: A generalized delta learning rule. Byte (October 1987) . 155-162 pp.

Kahn, G., Nowlan, S., and McDermott, J. 1985. MORE: An intelligent knowledge acquisition tool. Pages 581-584 in Proc. IJCAI-85, Los Angeles, CA.

Kandel, E.R., Schwartz, J.H. 1985. Principles of neural science. New York: Elsevier.

Kohonen, T. 1988. Self-organization and associative memory (2nd ed.). Berlin: Springer-Verlag.

Kohonen, T., Masisara, K., and Saramaki, T. 1984. Phonotopic maps—Insightful representation of phonological features for speech representation. A chapter in Proceedings IEEE 7th Inter. Conf. on Pattern Recognition, Montreal, Canada.

Kohonen, T. 1984. Self-organization and associative memory. Berlin: Springer-Verlag.

Korn, G.A., and Korn, T.M. 1964. Electronic analog and hybrid computers. New York: McGraw-Hill.

Kunz, J.C., Fallat, R.J., McClung, D.H., Osborne, J.J., Votteri, R.A., Nii, H.P., Aikens, J.S., Fagan, L.M., and Feigenbaum, E.A. 1978. A physiological rule-based system for interpreting Pulmonary function test results. HPP-78-19, Computer Science Department. Stanford, California: Stanford University.

Lee, W., and Ray, S. 1986. Rule refinement using the probabilistic rule generator. Pages 442-447 in Proc. AAAI-86, Philadelphia, **PA.**

Lidsay, R.K., Buchanan, B.G., Feigenbaum, E.A., and Lederberg, J. 1980. Applications of artificial intelligence for chemical inference: The DENDRAL project. New York: McGraw-Hill.

Linsker, **R. 1988.** Self-organization in a perceptual network. Computer (March 1988). 105-117 pp.

Lippman, R.P. 1987. An introduction to computing with neural nets. IEEE ASSP Magazine, April 1987.

Lippmann, R.P., Gold, B., and Malpass, M.L. (in press). A comparison of Hamming and Hopfield neural nets for pattern classification. MIT Lincoln Laboratory Technical Report, TR-769.

Martin, W.A., and Fateman, R.J. 1971. The MACSYMA system. A chapter in Proceedings of the Second Symposium **on** Symbolic **and** Algebraic Manipulation, Los Angeles, California.

McCulloch, W.S., and Pitts, W.A. 1943. A logical calculus of **the** ideas immanent in nervous activity. Bulletin of Mathematics **and** Biophysics 5:15-133.

McCulloh, W.S., and Pitts, W.A. 1943. A logical calculus **of** the ideas imminent in nervous activity. Bulletin of Mathematical Biophysics, Vol. 5. 115-133 pp.

McDermott, J. 1981. R1: The formative years. AI Magazine 2, No. 2.

Michalski, R.S., Carbonnell, J.G., and Mitchell, T.M. (eds.). 1983. Machine Learning (Tioga, Palo Alto, CA) .

Minsky, M., **and Papert**, S. **1969.** Perceptrons. Cambridge, MA: MIT Press.

Mishkoff, H.C., 1985. Understanding Artificial Intelligence. Harvard W. Sams & Co., Indianpolis, USA.

Mitchell, T.M., Keller, R.M., and Kedar Cabelli, S.T. 1986. Explanation-based generalization: A unifying view. Machine Learning l(l):47-80.

Moller, A.R. 1983. Auditory physiology. New York: Academic Press.
Mui, L. , and Agarwal,A. 1993. An Adaptive Modular Neural Network with Application to unconstrained Character Recognition. Discussion Paper. Cambridge, MA, USA: Massachusetts Institute of Technology.

Nagendraprasad, M.V., Liu,A. and Gupta,A. 1992. A System for Automatic Recognition of Totally Unconstrained Handwritten Numerals. IFSRC NO 218-92. Cambridge, MA, USA: Masachusetts Institute of Technology.

Nemes, T.N. 1969. Kibernetikai Gepek (Cybernetic Machines). Budapest: Akademiai Kiado.

Politakis, P.G. 1982. Using empirical analysis to refine expert system knowledge-base. Ph.D. thesis. USA: Rutger University.

Pople, H.E. Jr., Myers, R.D., and Miller, R.A. 1975. DIALOG: A model of diagnostic logic for internal medicine. International Joint Conferences on Artificial Intelligence 4.

Quinlan, J.R. 1987. Simplifying decision trees. Internat. J. Man-Machine Studies 27:221-234.

Rada, R. 1985. Gradualness facilitates knowledge refinement. IEEE Trans. Pattern Analysis and Machine Intelligence 7(5):123-128.

Rosenblatt, R. 1959. Principles of Neurodynamics. New York: Sparten Books.

Rosenblatt, F. 1958. The perception: A probabilistic model for information storage and organization in the brain. Psychoanalytic Review 65:386-408.

Rumelhart, D.E., Hinton, G.E., **and** Williams, **R.J. 1986.** Learning internal representation by error propogation. A chapter in Parallel Distributed Processing Explorations in the Microstructure of Cognition. USA: MIT Press.

Rumhelhart, D.E., and McClelland, J.L. **1986.** Parallel distributed processing: Explorations **in** the microstructure of cognition. USA: MIT Press.

Sejnowski, T.J., and Rosenberg, C.R. 1986. NETalk: A parallel network that learns to read aloud. JHU/EECS-86 01. USA: Johns Hopkins University.

Sejnowski, T., and Rosenberg, C.R. 1986. NETalk: A parallel network that learns to read aloud. Johns Hopkins Univ. Technical Report, JHU/EECS-86/01.

Shavlik, J.W., and Towell, G.G. **1989.** An approach to combining explanation-based and neural learning algorithms, Connection Science, Vol. 1 (3) : 231-252.

Shortliffe, E.H. 1976. Computer-based medical consultation: MYCIN. New York: Elsevier.

Smith, **R.G.,** Winston, H.A., Mitchell, **T.M.,** and Buchanan, B.G. **1985.** Representation and use of explicit justifications for knowledge-base refinement. Pages 673-680 in Proc. IJCAI-85, Los Angeles, CA.

Smolensky, P. 1987. Connectionist AI, symbolic AI, and the brain. Artif. Intell, Rev. Vol 1:95-109.

Steinbuch, K. 1961. Die Lernmatrix. Keybernetik 1:36-45.

Suwa, M., Scott, A.C., and **Shortliffe,** E.H. **1984.** Completeness and consistency in a rule-based expert system. A chapter in Rule-based Expert Systems (Buchanan, B.G., and Shortliffe, E.H., (eds.). USA: Addison-Wesley.

Towell, G.G., Shavlik, J.W., and Moordewier, M.O. 1990. Refinement of approximate domain theories by knowledge-based neural networks. A chapter in Proc. of Eight National Conference on Artificial Intelligence, AAAI-90.

Tsypkin, Y.A. 1968. Adaptation and learning in cybernetic systems. Moscow: Nauka.

Valtorta, M., 1989. "Some results on the complexity of knowledge-base refinement" in Proceedings of IWML-6. Los Altos, California: Morgan Kaufmann.

Wallace,D.J. 1986. Memory and Learning in a class of Neural Model in Proceddings of the Workshop on Lattice Gauge theory (Bunk.B., and Mutter K.H. eds). Wuppertal.

Wang,P.S.P. (ed). 1991. Character and Handwriting Recognition - Expanding Fronteirs. World Scientific Publishing.

Wasserman. 1989. Neural computing: Theory and practice. New York: Van Nostrand Reinhold.

Weiss, S.M., Kulikowski, **C.A. and** Safir, **A. 1978.** A model-based consultation system for the long-term management of Glaucoma. International Joint Conferences on Artificial Intelligence 5.

Widrow, B., and Hoff, M.E. 1960. Adaptive switching circuits. 1960 WESCON Convention, Record Part IV. 96-104 pp.

Wilkins, D.C., and Buchanan, B.G. 1986. On debugging rule sets when reasoning under uncertainty. A chapter in Proc. AAAI-86. USA: Philadelphia.

Yasaswy,J.J. 1990. Personal Investment and Tax Planning. New Delhi, India: Vision Books.