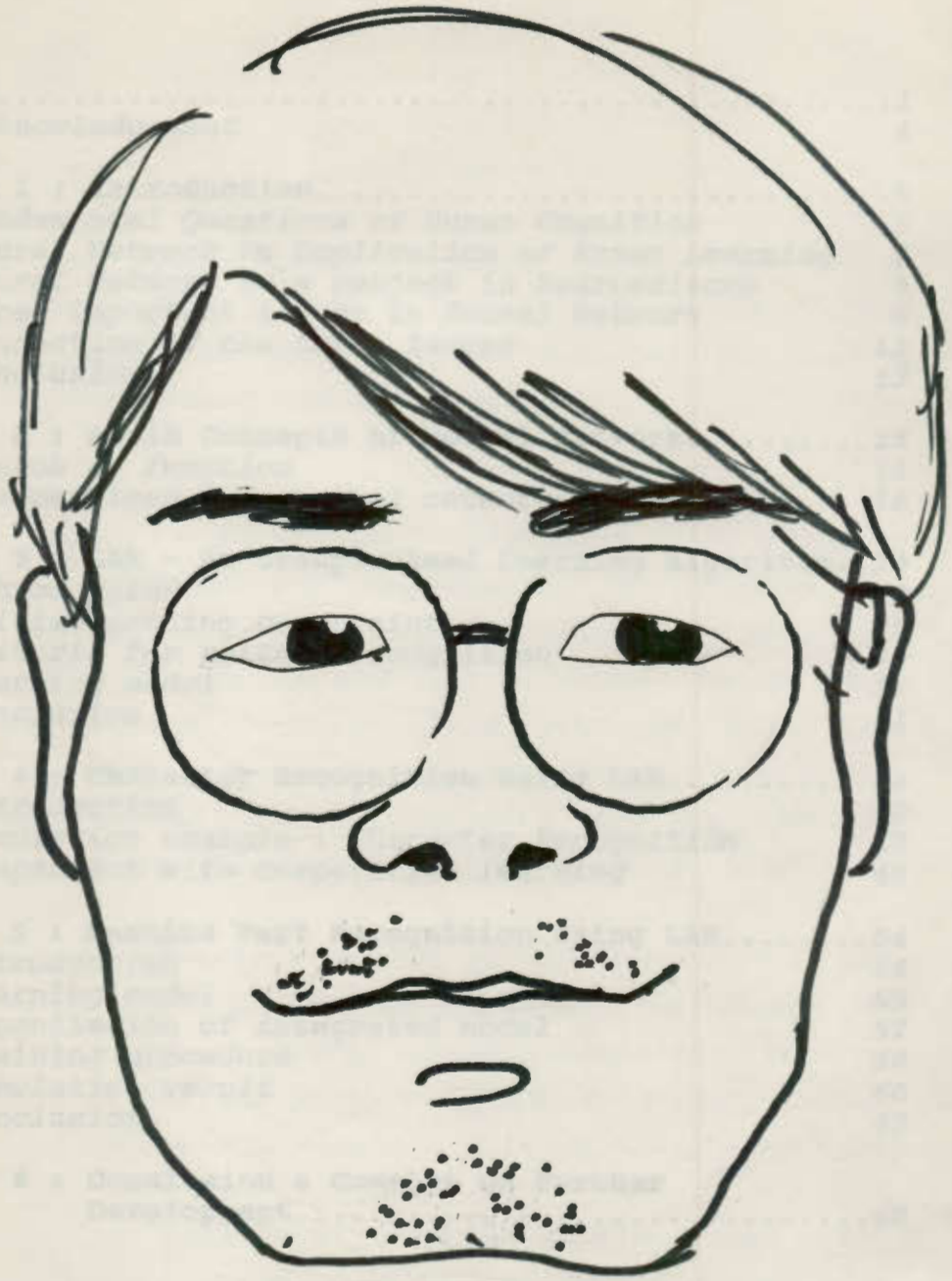
NEURAL NETWORK FOR CHARACTER RECOGNITION

The Study of Aritificial Neural Network and the Implementation of LAR in Character/Machine-Part Recognition

John P.F.SUM
Department of Electronic Engineering
Hong Kong Polytechnic

Supervised by : Peter K.S.TAM
Senior Lecturer
Section of Control and Instrumentation
Department of Electronic Engineering
Hong Kong Polytechnic



JOHN SUM. 1967- .....

CONTENTS

PREFACE

In the recent decade, the research in neural network came into a new era. Some researchers proposed new models of neural networks. Some enhanced the old ones with new structures, in order to reduce the complexity of the networks. New learning algorithms were proposed and investigated. Centers related to neural network research were set up in many university across the United State and European. Companies were also established for neural network research, development and application.

So far, a number of neural networks has been well developed and applied. The Multi-Layer Perceptron with Back Propagation learning algorithm was implemented in pattern recognition application successfully. In another school, the Adaptive Resonance Theory (ART), was applied not just in pattern recognition, but also in speech recognition. Hopfield Net was well developed as associative memory. Besides, the capability of VLSI fabrication has been deeply investigated. Futhermore, the application of neural network in system control is also becoming popular.

The scope of neural network is very large. It is impossible to give a full account, both on the historical background and the theory background, on neural network within this technical report. Hence, as a final year project report, only part of the theory and history, of the neural network, will be elucidated in the text.

Basically, neural networks can be classified into supervised and unsupervised types. Inside the report, most of the neural network models described will be in the category of unsupervised learning. There are two reasons why the writer preferred the unsupervised instead of the supervised type.

(1) Since the supervised type of neural networks was already studied and implemented in one of the final year projects in 1990. The writer do not want to overlap the job with that project. The interested reader can refer to that project report, written by C.W. Cheung in the 1990 (Cheung 90), to get the background on the supervised type of neural network. Besides, several models in supervised learning were also studied indepth in that project. Multi-Layer Perceptron was even implemented by him in Tactile image recognition and robotics motion control and the result was good.

(2) The core of this project is in the development and implementation of a novel neural network model, the LAR (Learning by Attraction and Repulsion) model. This model was designed by the writer early in 1992. The structure of LAR model is basically inspired from the Competitive Learning one which is an unsupervised learning model. As an introductory section, most of the models described are of the unsupervised type.

Priori to the main content, an **introduction** is given in chapter one. It gives a concise historical background on neural network, its appearance, its relationship with other subjects and its role. It is aimed at providing an overview for the reader. So, it acts as a bridge for the new comer. Since the presentation approach may not be good, if the reader get lost in this chapter, he can skip this chapter and find another way to get the historical information. Let me emphasis, background information may not help the reader to understand the theory of the neural network but it can give the reader a sense of completeness in this area. One suggested reading is Principle of Neurodynamics. It was written by Frank Rosenblatt in 1962 (Rosenblatt 1962). In chapter 3 of the book, Frank Rosenblatt described clearly the neural network history including the arising of neural network and its relationship with psychology, physiology and etc.

In chapter two, the **basic concept of theory of neural network** will be presented. As mentioned before, it is mainly unsupervised type, including Hopfield Net, Competitive Learning and ART. The reader who wishes to have a deeper insight on the supervised model can refer to "The Study of the Applications of Neural Networks" (Cheung 1990), "Neurocomputing" (Hecht-Neilsen 1990), "Neural Network and Fuzzy Systems" (Kosko 1992) or "Neural Computing" (Wasserman 1989). Besides, the writer presumes that the reader have acquired basic knowledge on the neuron structure. For further details, the reader can refer to chapter 15 of "Biopsychology" written by John Pinel (Pinel 1990).

The **LAR model** is given in chapter three. It was actually the core stuff in the final year project. A number of simulation programs and application programs have been written for the verification and evaluation of the model. Part of the result will also be provided for the explanation. The LAR model was also implemented in **character recognition** and **machine part recognition**. Details of these applications will be found in chapter four and five respectively.

In chapter six, the **further development** on the LAR model will be presented. A number of Appendices will follow. This materials serve as a source for the reader. When the reader find problem in understanding the algorithm or the content of the report, he can refer to the appendices. At the end of the report, a list of reference will be provided in order to let the reader to trace the sources of information, quoted inside the report.

In summary, the objective of this report consists of the following five points.

1. It introduces the historical background of neural networks.
2. It introduces the basic ideas of neural network.

3. It explains the theory of the LAR model.
4. It provides the sources of information.
5. It provides the programs and data files for the LAR model.

Most important, the writer wishes that the report can also be a pointer for the new comer.

ACKNOWLEDGEMENT

Finally, I would like to thank some people who helped me in completing this project. The first two are Mr.Lee Lin Fai, Electronics Engineering Degree Four, and Miss Florence Tang, Language and Communication Degree Two. Both of them gave me a lot of advises during the development of the LAR model. Discussion with them was worthwhile and stimulative. The second two whom I would like to thank are Dr. Peter Tam (my project supervisor) and Dr. C.K. Lee, Department of Electronic Engineering, HKP. Without their valuable suggestion and encouragement, the LAR model might not be appeared so early and the papers might not be submitted so easy. Most of all, I would like to thank my uncle Anthony To, Master Candidate, Department of Educational Psychology, University of Hong Kong. Without his provision of the information in psychology, I will surely miss plenty of important aspects of the neural network. Besides, I would like to give a special thank to Dr. Peter Tam for his patient reviewing on the manuscript. Of course, the writer is willing to response to any mistake appeared in this report. I apologize to those of my friends whose contribution I cannot include in this brief acknowledgment here. I am sorry.

John P.F. SUM
(Degree 4) Department of Electronics Engineering,
Hong Kong Polytechincs. HONG KONG.
May 1992

CHAPTER ONE : INTRODUCTION

In this chapter, some important ideas on the neural network will be given. Its connection with other fields of studies, such as artificial intelligence, brain theory, psychology etc,. will be examined. Actually, half of the content of this chapter is summarized from Frank Rosenblatt (1958), Rosenblatt (1960), Rosenblatt (1962) and Rosenblatt (1964).

Fundamental Questions of Human Cognition

If we are eventually to understand the capability of higher organisms for the perceptual recognition, generalization, recall and thinking, we must first have answers to the following three fundamental questions:

1. How is information about the physical world sensed, or detected, by a biological system?

2. In what form is the information stored, or remembered?

3. How does information stored or remembered influence recognition and behavior.

The first of these questions is in the province of sensory physiology, and is the only one for which appreciable understanding has been achieved. With regard to the second question, two alternative positions have been maintained.

The first suggests that storage of sensory information is in the form of coded representation of images, with same sort of one-to-one mapping between sensory stimulus and the stored pattern. According to this hypothesis, if one understood the code of the nervous system, one should in principle be able to discover exactly

what an organism remembers by reconstructing the original sensory patterns from the "memory traces".

The alternative approach which stems from the tradition of British empiricism, hazards the guess that images of stimuli may never really record at all, and that the central nervous system simply act as an intricate switching network, where retention takes the form of new connections, or pathways, between centers of activity. The important feature of this approach is that there is never any simple mapping of the stimulus into memory, according to some code which would permit its later reconstruction.

Corresponding to these two positions on the method of information retention, there exist two hypotheses with regard to the third question. The "code memory theorists" are forced to conclude that recognition of any stimulus involves the matching of systematic comparison of the contents of storage with incoming sensory patterns. The theorists in the empiricist tradition have essentially combined the answer to the third question with their answer to the second: since the stored information takes the form of new connections, or transmission channels in the neurons system, it follows that the new stimuli will make use of these new pathways which have been created, automatically activating the appropriate response without requiring any separate process for their recognition or identification. The theory backing the Perceptron and neutral network takes the empiricist or connectionist position.

At this moment, the reader should realize that in the recent decades, "code memory theorists" was the symbolic approach. It played a main role in the study of artificial intelligence research.

Neural Network Vs Duplication of Human Learning

The following message is also quoted from Rosenblatt (1960) while he was answering a question from an audience. In his answer, he explained clearly his attitude towards his research area. It should be the attitude of neural networks research workers nowadays.

Well, first of all let me say that we are interested in duplicating human learning, if it is possible to do so. We are interested in determining the extend to which it is feasible to consider such a thing as duplicating human learning, or at least understanding how human learning operates. Whether or not there exists a better mode of learning is in sense an empirical question to which I don't feel we can supply an answer at this point.

We interested, however, not only in studying human learning, but in studying the behavior of networks which include biological nervous systems as a subclass. That is to say, we are interested in study of signal transmission networks which involve connected nodes or cell points which have functional characteristics similar to those of biological neurons, but not necessarily identical.

If it emerges from the study of such systems that some of these behave better than others or some of them so in fact behave better than human nervous systems, this would be a very interesting finding indeed. But it would emerge from the study of this general class of systems and is not something I feel we can specially aim for at this point.

Neural network as a subject in neuroscience

Neuroscience can be broadly defined as a scientific displince concerned with understanding both the brain and the mind, which are usually presumed to be respectively the hardware and software aspects of the same object. In neuroscience, the brain's structure has been simulated by creating many functional concepts and psychological models based on experimental results. Many valuable inspirations provide insight to neural network. As a result, many models have been created, which are simplified versions of the actual human brain. In return, the ideas evolved from neural network provide useful insights to the research of neuroscience (Rosenblatt et al, 1966; Rosenblatt 1967; Kandel 1982; Pinel 1990; Churchland et al 1990).

Three important issues in Neural Network

In the history of neural network, there are three issues paving the way for the current research. One was the neuron model proposed by Warren MuCulloch and Walter Pitts in 1943 (MuCulloch and Pitts 1943). The other was the Hebb's postulation, or Hebbian Learning, which was claimed in 1949 (Hebb 1972; Hebb 1980). M-P neuron may be represented as shown in Fig 1.1(b) and in its simplest possible form is a device which gives an output (to the right) if it gets an input from at least a certain number, say θ, of its inputs (on the left). It thus has a threshold θ in the simplest version of the model. Biologists often criticize the logical neuron for being too unrealistic. It is important for us to realize that this is rather unfair. The great advantage of the logical neuron is its simplicity, which often enables us easily to gain an insight into how a network of nerve cells might be expected to behave. The last issue was the derivation of the membrane equation by A. Hodgkin and A. Huxley in the 1952 (Hodgkin and Huxley, 1952).

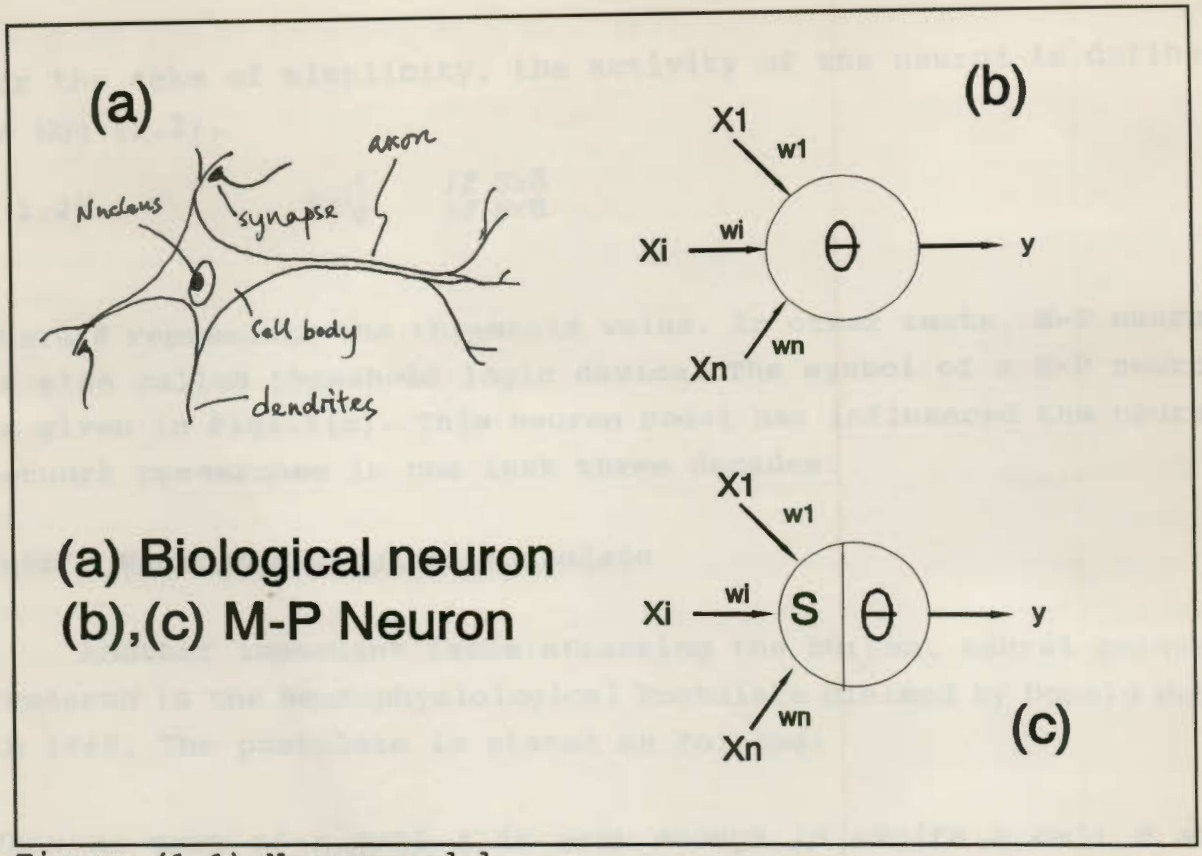


Figure (1.1) Neuron model

MuCulloch and Pitts Neuron (MuCulloch and Pitts 1943; Griffith 1971)

In Fig 1.1(b), x_i is the output signal from the ith neuron. To be more precise, it is the potential at the axon of the ith neuron. w_i is the synaptic strength between the ith neuron and the neuron y. Denote the effective potential arrived at the dendrite of the neuron as S. The effective potential, S, is given by Equ(1.1).

(1.1) $$S = \sum_{i=1}^{n} w_i x_i$$

For the sake of simplicity, the activity of the neuron is defined as Equ (1.2).

$$y=\begin{cases}1 & \text{if } S\geq\theta \\ 0 & \text{if } S<\theta\end{cases} \tag{1.2}$$

where θ represents the threshold value. In other texts, M-P neuron is also called threshold logic device. The symbol of a M-P neuron is given in Fig1.1(c). This neuron model has influenced the neural network researches in the last three decades.

Hebb's Neurophysiological Postulate

Another important issue affecting the current neural network research is the Neurophysiological Postulate claimed by Donald Hebb in 1949. The postulate is stated as follows:

When an axon of a cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B is increased.

Donald Hebb also suggested that one cell could become more capable of firing another is due to the synaptic knobs development (Hebb 1949). The mathematical representation of the postulate is given by Equ(1.3).

$$\frac{\partial}{\partial t}w_i-\alpha\geq 0 \tag{1.3}$$

According to the postulate, the strength of the synapse will be increased if the postsynaptic cell and presynaptic cell are firing simultaneously or repeatedly. Besides, a collolary on the Hebb's postulation is also suggested in parallel. The mathematical

interpretation is given by Equ(1.4).

$$\frac{\partial}{\partial t} w_i - \alpha < 0 \qquad (1.4)$$

When an axon of cell A is near enough to cell B and repeatedly firing B, A's efficiency is decreased if B is not be excited.

Hodgkin-Huxley Membrane Equation

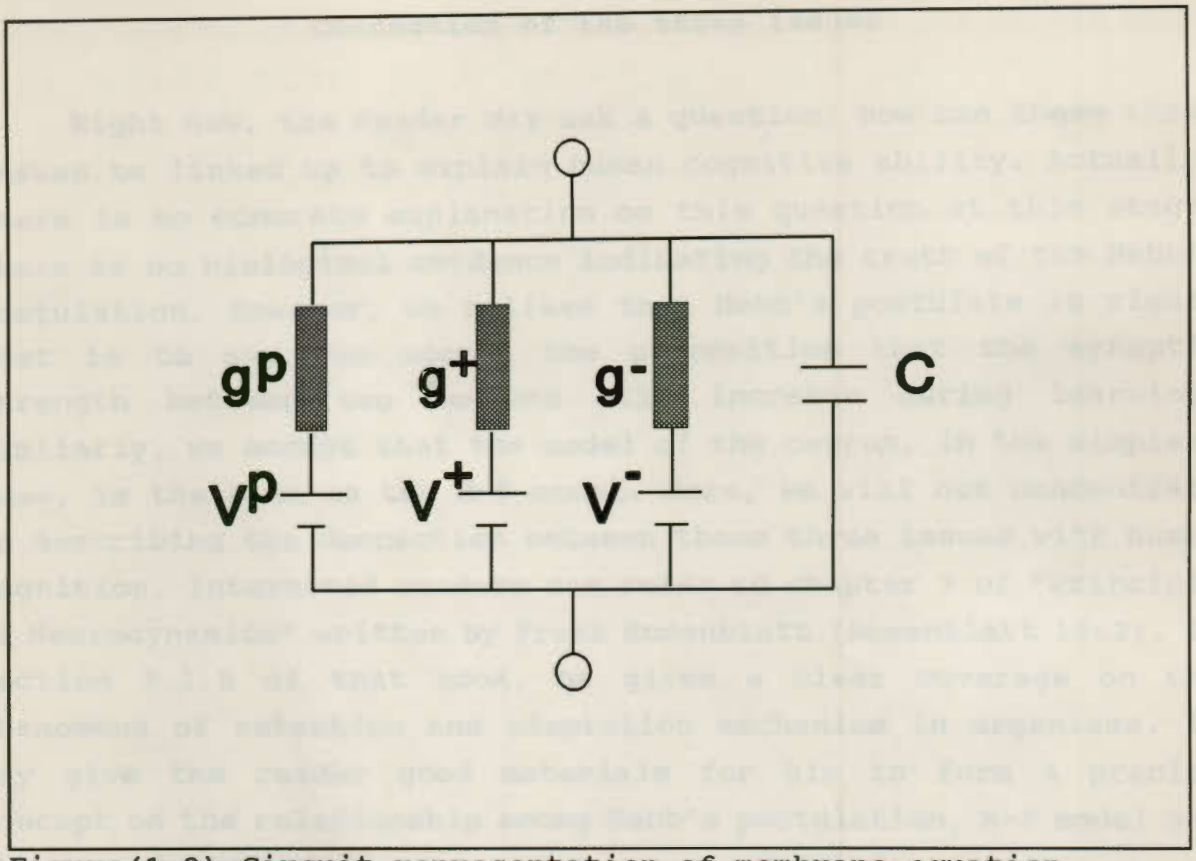
Nerve cells generate electrical signals by gating the ions channels. The ability of nerve cells to gate their ion channels allows them to control the permeability of their membranes and to regulate the diffusion of selected ions down preestablished electrochemical gradients. Consider the properties of the ion channels used for signalling in electrical terms, a simple mathematical model was derived, in the form of Equ(1.5).

$$C\frac{\partial V}{\partial t} = (V^p - V) g^p + (V^+ - V) g^+ + (V^- - V) g^- \qquad (1.5)$$

V^p, V^+ and V^- denote respectively passive (chloride CL^-), excitatory (sodium Na^+), and inhibitory (potassium K^+) saturation upper bounds with corresponding shunting conductance g^p, g^+ and g^-. C is the capacitance. Figure(1.2) shows the equivalent electric circuitry.

At equilibrium the Hodgkin-Huxley model has the resting potential V_{rest}, given by Equ(1.6). While the chloride-based passive terms are neglected, the resting potential is given by Equ(1.7).

$$V_{rest} = \frac{g^p V^p + g^+ V^+ + g^- V^-}{g^p + g^+ + g^-} \qquad (1.6)$$



Figure(1.2) Circuit representation of membrane equation

(1.7) $$V_{rest}=\frac{g^{+}V^{+}+g^{-}V^{-}}{g^{+}+g^{-}}$$

Though the equation was proposed in 1952, utilization of this equation in the neural network model was not usual. Most of the neural network models proposed in the 1960s or earlier made use of the McCulloch-Pitts neuron model. After 1960s, only a limited number of researchers, such as Stephen Grossberg and his colleagues, made use of this membrane equation in the neural network models. Michael Cohen and Stephen Grossberg further proved that those neural networks described by the relationship in the form of Equ(1.5) can have a global stability (Cohen and Grossberg 1983). Some writers call this Cohen-Grossberg Stability Theorem (Kosko 1992).

Connection of the three issues

Right now, the reader may ask a question: how can these three issues be linked up to explain human cognitive ability. Actually, there is no concrete explanation on this question at this stage. There is no biological evidence indicating the truth of the Hebb's postulation. However, we believe that Hebb's postulate is right. That is to say, we accept the proposition that the synaptic strength between two neurons will increase during learning. Similarly, we accept that the model of the neuron, in the simplest case, is the same as the M-P model. Here, we will not concentrate on describing the connection between these three issues with human cognition. Interested readers can refer to chapter 3 of "Principle of Neurodynamics" written by Frank Rosenblatt (Rosenblatt 1962). In section 3.1.5 of that book, he gives a clear coverage on the phenomena of retention and adaptation mechanism in organisms. It may give the reader good materials for him to form a precise concept on the relationship among Hebb's postulation, M-P model and the neural network.

Conclusion

Neural network is a hot research topic Some people even claimed that neural network architecture had the potential to realize a new type of computer, replacing the traditional computer. However, this goal is still very far from reachable. Actually, what will be the fate of neural network research is still a question. Will it be the same as in the 1970s when neural network research was in a silent period? The answer is still not known.

In so far, the writer has tried to describe a very brief overview on certain aspects of the neural network, including its historical background and its relationships with other subjects. But the story of neural network is much more intricative and much

longer than represented on these few pages. It is out of the scope of a project report to spell out the whole history of neural network and it is also out of the scope of the ability of the writer.

The following chapters will not touch on history. Instead, they will concentrate on the concept and theory of unsupervised learning models, especially the LAR model (Lee et al 1992a).

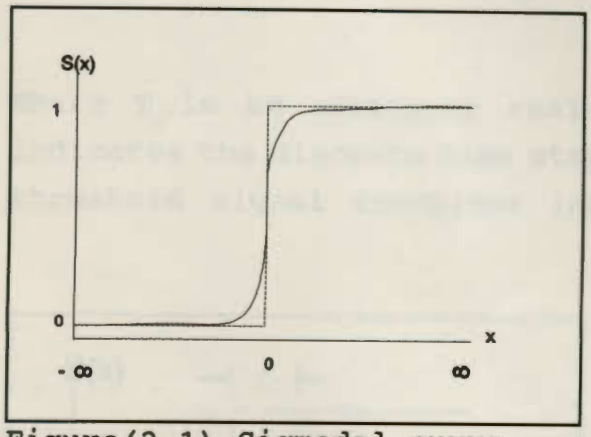
CHAPTER 2 : BASIC CONCEPT OF NEURAL NETWORK

As mentioned in chapter one, in this chapter we will present a description of the basic concepts concept on some aspects of neural network aspects and several unsupervised network models. Those reader who find the content being too brief can refer to the reference listed for further information.

In this chapter, we first model the neuron as function and introduce a few common signal functions. After that, three neural models will be presented including the Hopfield Net, Competitive Learning, and ART.

Neuron As Function

Neurons behave as functions. Neurons transduce an unbounded input activation $x(t)$ at time t into a bounded output signal $S(x(t))$. Usually a sigmoidal or S-shaped curve, as in Figure 2.1, describes the transduction.



Figure(2.1) Sigmodal curve

In general, there are several common types of neuron functions used in the current neural network research. In the following text, only six of them will be given. For more details, reader can refer to chapter two of Kosko's book (Kosko 1992).

1. Logistic Signal Function

It is the most popular binary signal function. We define the function as Equ(2.1). The shape of this signal function can be

refer to Fig.(2.1).

$$S(x) = \frac{1}{1 + e^{-cx}} \tag{2.1}$$

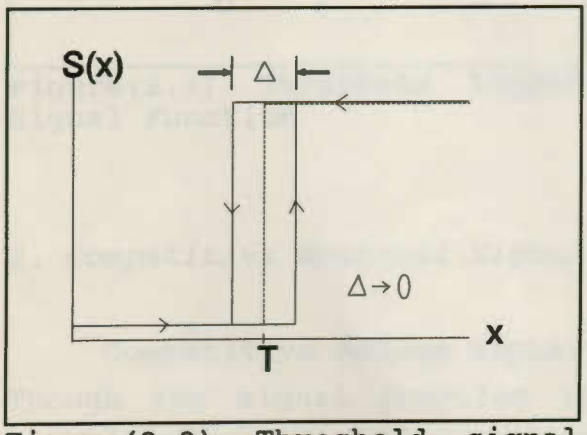
Where c is a positive number. Obviously, the function is a monotonically increasing function. Nearly all supervised type neural networks, such as Adaline and Back-propagation perception, use this function, logistic signal function, as the neuron transfer function. Sometimes we also call this function as sigmoid function.

2. Threshold Signal Function

Another common neuron function is the threshold signal function. Actually, is an infinitely steep logistic signal function. The characteristic equation is given by Equ(2.2).

$$S(x^{k+1}) = \begin{cases} 1 & \text{if } x^{k+1} > T \\ S(x^k) & \text{if } x^{k+1} = T \\ 0 & \text{if } x^{k+1} < T \end{cases} \tag{2.2}$$

Where T is an arbitrary real-value threshold T. The index k indicates the discrete time step. The index notation implies that threshold signal functions instantaneously transduce discrete activations to signals. When the activation equals the threshold at time k+1, the signal maintain the same value it had at time k. The neuron does not make a state-update "decision" at time k+1. This type of neuron function appears in one of the core neural network nowadays, the Hopfield Network (Hopfield 1982; Hopfield 1984). Figure 2.2 shows the shape of the threshold



Figure(2.2) Threshold signal function

signal function.

3. Hyperbolic-tangent signal Function

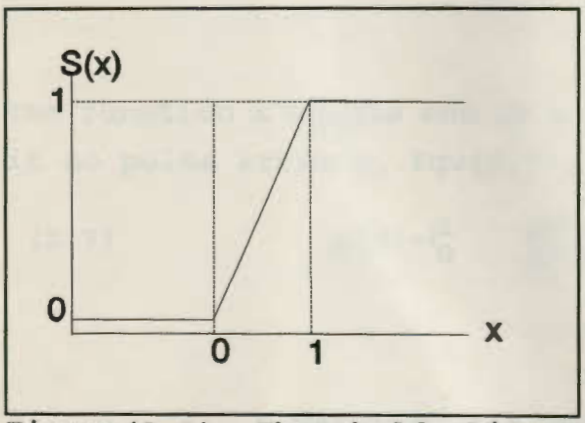
It is a naturally occurring bipolar signal function. It shape looks like Figure 2.1. The transfer characteristic is given by equation Equ (2.3).

(2.3) $$S(x)=\tanh(cx)$$

This signal function is also very commonly used in many supervised learning models such as Adaline and backpropagation.

4. Threshold Linear Signal Function

Threshold linear signal function is a binary signal function often used to approximate neuronal firing behaviour. The function is given by Equ(2.4) and the shape is indicated by Fig(2.3).



Figure(2.3) Threshold Linear Signal Function

(2.4) $$S(x)=\begin{cases}1 & \text{if } cx\geq 1\\ 0 & \text{if } 0<cx<1\end{cases}$$

and S(x)=0 if cx<0. Obviously, it is also a monotonically increasing function.

5. Competitive Neuronal Signal

Competitive Neuron Signal is used in competitive learning. Though the signal function is simple, its mechanism is very complicated (Lippmann 1988). The effect is due to the lateral inhibitory connection within the output layer. Anyway, as a

transfer function, we are not concerned with the mechanism. The equation of competitive signal function is given by Equ(2.5).

$$S(x_j) = \begin{cases} 1 & \text{if } x_j = \max\{x_j\} \\ 0 & \text{else} \end{cases} \tag{2.5}$$

This equation is very simple. Competitive learning is based on this equation.

6. Pulse-Coded Signal Function

Pulse-Coded signal function is recently proposed by Bart Kosko (Kosko 1992). In the field of neuron research, scientist have already recognize the pulse trains propagation phenomenon. However, this idea has not been used. Kosko and Kong recently built a number of neural network models, such as differential competitive learning model and differential Hebbian learning models (Kong and Kosko 1991; Kosko 1990; Kosko 1991), based on this signal function. The function is given by Equ(2.6).

$$S(t) = \int_{-\infty}^{t} x(u)\, e^{u-t} ds \tag{2.6}$$

The function x equals one if a pulse arrives at time t, and zero if no pulse arrives, Equ(2.7).

$$x(t) = \begin{cases} 1 & \text{if a pulse occur at } t \\ 0 & \text{if no pulse at } t \end{cases} \tag{2.7}$$

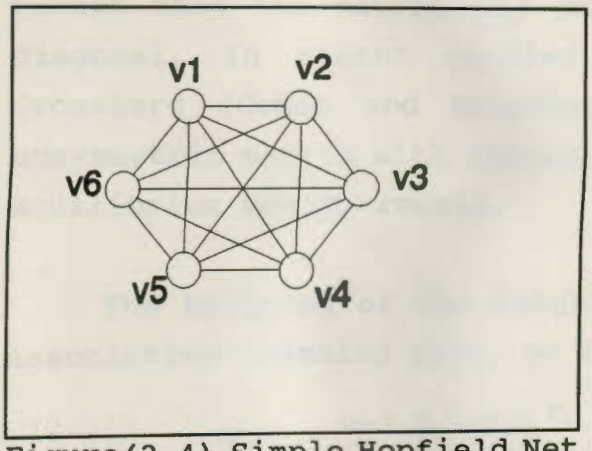
Unsupervised Type Neural Networks

The development of unsupervised learning algorithms has been long in history (Rosenblatt 1958; McCleland and Rumelhart 1981; Rumelhart and McCleland 1982; Rumelhart and Zipser 1985). It can be traced back to 1958, when the Gamma Perceptron was invented by Frank Rosenblatt. Competitive Learning is one of the simplest paradigm in this area. Though competitive learning can be applied in vector quantization, clustering and categorization of

patterns, its application in pattern recognition is very poor. Stephen Grossberg slightly mentioned in his 1987 paper (Grossberg 1987). But he did not give reason for this incapability.

In the last decade, Stephen Grossberg reclaimed the Adaptive Resonance Theory and he also proved that ART can eliminate this limitation (Grossberg 1987). Grossberg and his colleagues implemented this model in adaptive pattern recognition (Carpenter and Grossberg 1987a, Carpenter and Grossberg 1987b). In the following section, three unsupervised neural networks will be introduced. They are the Hopfield Network, developed by John Hopfield in CalTech, Competitive Learning model which has been sustained for more than three decades and the last one is ART.

1. Hopfield Network



Figure(2.4) Simple Hopfield Net

A simple Hopfield Net is shown in Fig(2.4). This structure is a special case of associative network, or recurrent associative network.

The weight values of Hopfield Network follow the restrictions imposed by Equ(2.8) and Equ(2.9).

(2.8) $$w_{ij}=w_{ji}$$

(2.9) $$w_{ii}=0$$

The Neuron[1], v_i has a transfer function being a threshold signal function, Equ(2.2).

---

[1] In some other articles, it is also called a processing element or a node.

$$S_i(x_i^{k+1}) = \begin{cases} 1 & \text{if } x_i^{k+1} > T_i \\ S_i(x_i^k) & \text{if } x_i^{k+1} = T_i \\ 0 & \text{if } x_i^{k+1} < T_i \end{cases} \tag{2.2}$$

where

$$x_i^{k+1} = \sum_{j=1}^{n} w_{ij} x_j^k \tag{2.10}$$

The neurons are updated one at a time. The stability proof of the Hopfield net can be found in a number of articles (Hopfield 1982, Hopfield 1984 and Kosko 1992).

The Hopfield network does not have a learning law associated with its transfer function. The weight matrix is specified in advance. No restriction on the real number values w_{ij} are made except that the matrix (w_{ij}) must be symmetric and have a zero diagonal. In recent studies by Michael Cohen and Stephen Grossberg (Cohen and Grossberg 1983), it is indicated that unsymmetric matrix with non-zero diagonal can also have a stable equilibrium memory recall.

The building of the weight matrix is usually followed the associative learning rule, or Hebbian learning, Equ(2.11).

$$W = X_1 X_1^T + X_2 X_2^T + \ldots + X_L X_L^T \tag{2.11}$$

where $\{X_1, X_2, \ldots, X_L\}$ is the set of the pattern vectors. If the set $\{X_1, X_2, \ldots, X_L\}$ is an orthonormal set, then the recall will be perfect. ie. when an input pattern $Y = X_k$ is input to the network and let it interate. If the pattern set is an orthonormal set, then Y will be given by Equ(2.12). If the set is not orthonormal, then correlation noise, η, will exist, Equ(2.13).

$$Y = WX_k = X_k \tag{2.12}$$

(2.13) $Y=X_k+\eta$

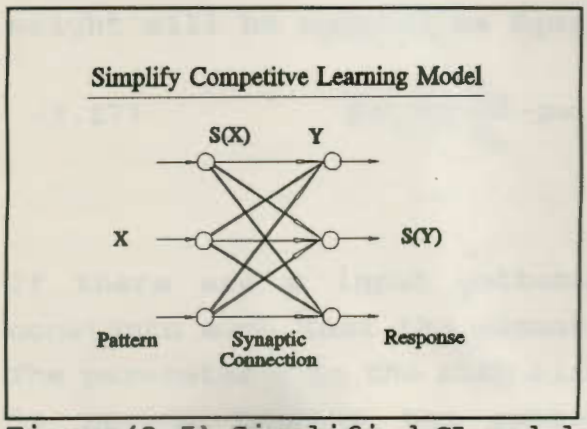
Theoretical proof has shown that the noise free memory capacity of a Hopfield net is 0.15n, (Hertz et al, 1991), where n is the number of neurons. Recently, C.C.Hui and L.W.Chan of the Chinese University of Hong Kong proposed ***An Error Correcting Algorithm for Hopfield Network*** (Hui 1991). This algorithm upgrades the noise free memory capacity of a Hopfield Network to 0.85n.

2. Competitive learning

Competitive Learning is one of the oldest neural network algorithms. Its history can be traced back to the 1958 (Rosenblatt 1958). In the next paragraph, this model will be elucidated.

Definition

Competitive learning model is simply a two layers neural network as shown in Fig(2.5). Each of the neurons in the first layer is connected to all the neurons in the second layer. Response of the first layer is based on the all-or-none principle. That is, when the neuron i in the first layer receives signal, x_i, is greater than a threshold, this neuron will deliver an impulse, $S_i(x_i)$, Equ(2.14).



Figure(2.5) Simplified CL model

(2.14) $$S_i(x_i)=\begin{cases}1 & if\ x_i>\theta_i \\ 0 & if\ x_i\le\theta_i\end{cases}$$

All impulses from the first layer will then pass to the second layer through the synaptic connections, and the strength of the synapse is denoted by w_{ji} where j represents the jth neuron in the second layer.

Winner-Takes-All Rule

The response of the second layer obeys the winner-takes-all rule. Suppose y_j is the potential of the jth neuron. The response of the neuron is given by Equ(2.15).

(2.15) $$S(y_j) = \begin{cases} 1 & \text{if } y_j = \max\{y_j\} \\ 0 & \text{else} \end{cases}$$

The value of y_j is actually the effective potential received from the first layer, given by Equ(2.16).

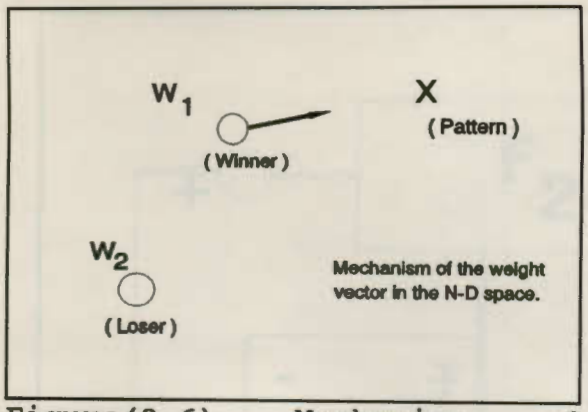
(2.16) $$y_j = \sum_{i=1}^{n} w_{ji} S_i(x_i)$$

Obviously, there is only one neuron which will be active, ie. S(y)=1. Once a neuron wins, say neuron j, its corresponding weight will be updated as Equ(2.17).

(2.17) $$\Delta w_{ji} = g\frac{C_{ik}}{n_k} - gw_{ji}$$

If there are m input patterns, k=1,2,....,m Cik and nk are constants such that the summation of the Cik/nk is equal to 1. The parameter g is the step size. Accordingly, the weight vector is moving towards the pattern vector, Fig(2.6). Generally speaking, we can state the principle of competitive learning as the following:

The winner will be given to the one which is the closest to the pattern, otherwise it will be a loser. For a winner, it will be attracted towards the pattern; otherwise, as a loser, it will remain unchanged at all.



Figure(2.6) Mechanism of competitive learning

In the next chapter, we will come back to the limitation of the competitive learning in applications in pattern recognition. In here, we simply state that the limitation is due to the initial setting constraint. Besides, the criteria for an unsupervised learning algorithm to recognize pattern will also present.

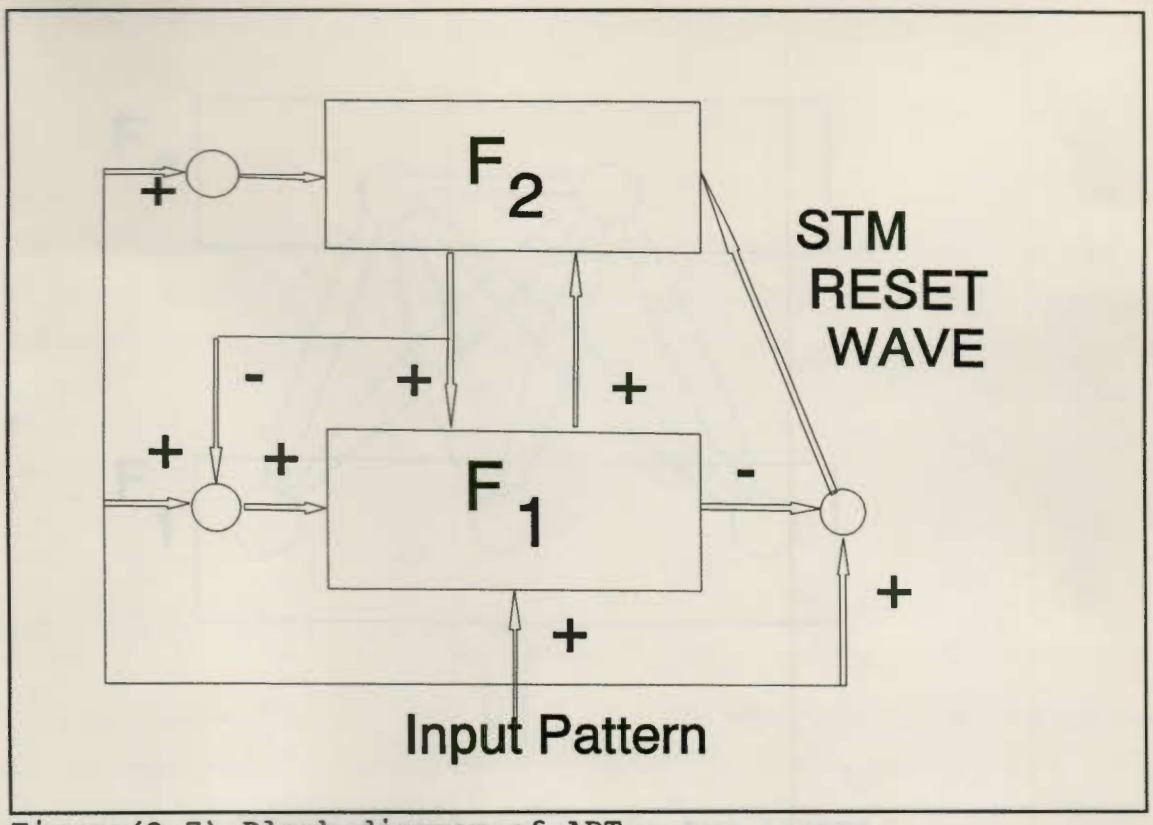
3. Adaptive Resonance Theory

ART network model is capable of self-organizing and self-stablizing its recognition codes in response to arbitrary temporal sequences of arbitrarily many input patterns of varible complexity. Due to its complicated structure, the details of the ART will not be elucidated here. Readers interested in can refer to Carpenter and Grossberg (1987a) and Grossberg (1987). Besides, Wasserman (1989) and Lippmann (1988) have also introduced ART, in a much simplified way.

The structure of an ART is shown in Fig(2.7). For simplicity, the LTM traces between F_1 and F_2 can be redrawn as shown in Fig(2.8). In this, there are only two neurons in the F_2 layer and three neurons in the F_1 layer. Moreover, the input pattern is restricted to be in binary form. Before giving the mechanism of the model, a list of definition is presented in the following.

Definition

1. F_1 activity pattern is given by $X=(X_1,X_2,\ldots,X_M)$.
2. F_2 activity pattern is given by $Y=(X_{M+1},X_{M+2},\ldots,X_N)$.
3. We denote nodes in F_1 by v_i, where $i=(1,2,\ldots,M)$ and nodes



Figure(2.7) Block diagram of ART

in F_2 by v_j, where j=(M+1,...,N).

4. The transfer function of the node in F_1 is given by h(x) and so the output pattern of F_1 is given by $S=(h(x_1),h(x_2),\ldots,h(x_M))$.

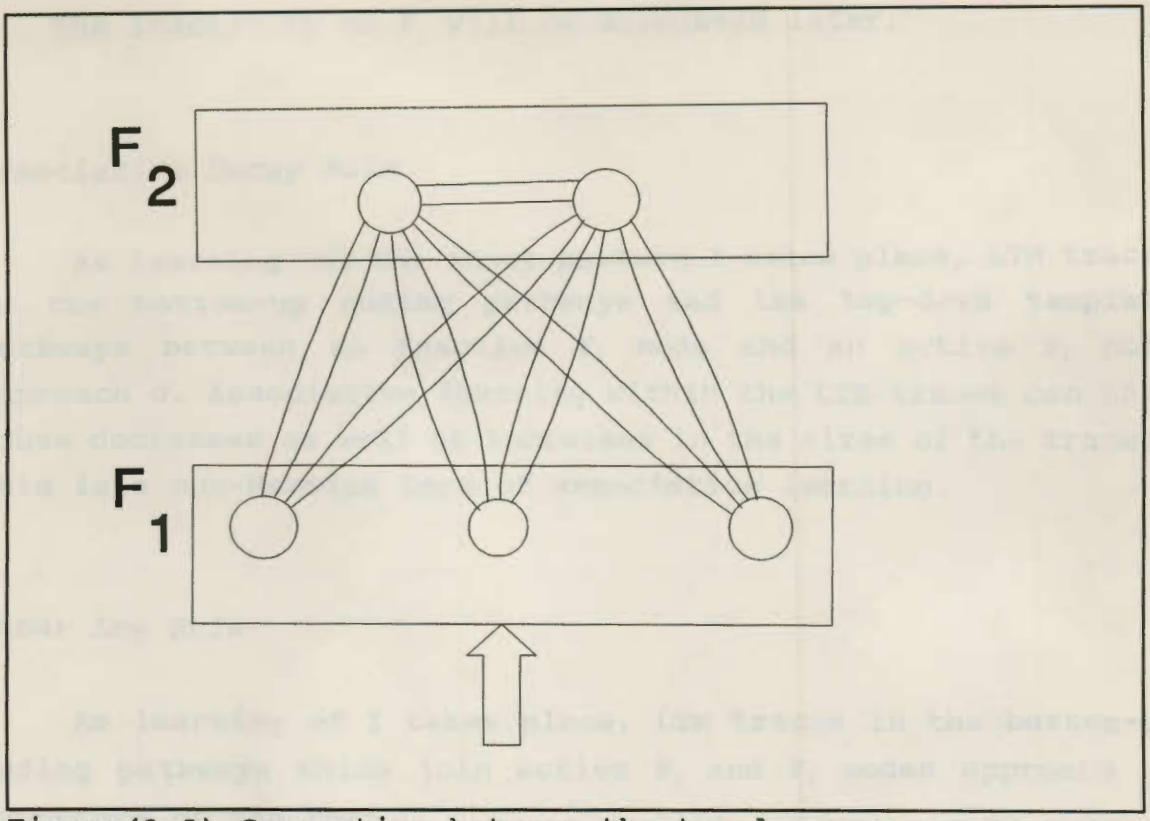
5. z_{ij} is defined as the (bottom-up) LTM trace, or called weight, in the pathway from v_i to v_j. Similarly, we define z_{ji} as the (top-down) LTM trace in the pathway from v_j to v_i.

6. The signal recieved by each of the nodes in F_2 is given by T_j, where j=(M+1,M+2,...,N), Equ(2.18). We also define $T=(T_{M+1},T_{M+2},\ldots,T_N)$ as the input to the F_2.

(2.18) $$T_j=\sum_{i=1}^{M} z_{ij}h(x_i)$$

7. The transfer function of the nodes in F_2 is given by f(x), Equ(2.19), and so the output pattern of F_2 is given by $V=(f(x_{M+1}),f(x_{M+2}),\ldots,f(x_N))$.

(2.19) $$f(x_j)=\begin{cases}1 & \text{if } T_j=\max\{T_k\}\\ 0 & \text{otherwise}\end{cases}$$



Figure(2.8) Connection between the two layers

8. The top-down template, recalled from F_2, is defined as $V=(V_1,V_2,\ldots,V_M)$, where V_i is given by Equ(2.20).

(2.20) $$V_i = \sum_{j=M+1}^{N} f(x_j)\, z_{ji}$$

Obviously, a different pattern U from F_2 will ignite a different pattern V. Since there is only one node in F_2 with output 1, we define $V^{(j)}$ as the top-down template due to node j in F_2.

9. The external input pattern is given by $I=(i_1,i_2,\ldots,i_M)$.

10. The activity of the layer F_1 is given by Equ(2.21).

(2.21) $$X = \begin{cases} I & \text{if } F_2 \text{ is inactive} \\ I \cap V^{(j)} & \text{if } F_2 \text{ node } v_j \text{ is active} \end{cases}$$

The inactivity of F_2 will be discussed later.

Associative Decay Rule

As learning of the input pattern I takes place, LTM traces in the **bottom-up coding pathways and the top-down template pathways** between an inactive F_1 node and an active F_2 node approach 0. Associative learning within the LTM traces can thus cause decreases as well as increases in the sizes of the traces. This is a non-Hebbian form of associative learning.

Weber Law Rule

As learning of I takes place, LTM traces in the **bottom-up coding pathways** which join active F_1 and F_2 nodes approach an asymptote of the form

$$(*) \qquad \frac{\alpha}{\beta+|I|} \quad \textit{where } \alpha, \beta \textit{ are positive constant}$$

Obviously, a larger norm of I, defined in Equ(2.26), implies a smaller positive LTM trace in the pathways encoding I.

Mechanism of the ART learning

For simplicity, the transfer function on the two layers are defined as Equ(2.22) and Equ(2.23).

(2.22) $\qquad h(x_i)-x_i$

(2.23) $\qquad f(x_j)-x_j$

Moreover, the top-down and bottom-up traces are initialized as Equ(2.24) and Equ(2.25), according to Lippmann (1988).

(2.24) $$z_{ji}(0) = 1$$

(2.25) $$z_{ij}(0) = \frac{1}{1+M}$$

When a new pattern is input to F_1, we let the network evaluate the top-down template V and X^*, following the sequence.

$$I \rightarrow X \rightarrow S \rightarrow T \rightarrow Y \rightarrow U \rightarrow V \rightarrow X^*$$

X^* is calculated by Equ(2.21). Here, we also define the norm of X as the number of ones, Equ(2.26).

(2.26) $$|X| = \sum_{i=1}^{M} x_i$$

Once X^* has been found, a vigilance test is applied as described by Equ(2.27).

(2.27) $$\frac{|X^*|}{|X|} > \rho \qquad where\ 0 \le \rho \le 1$$

If the result is true, then the corresponding weights will be updated. If the result is false, then the best matching node selected is tempararily disabled by the STM RESET WAVE and the top-down template is recalculated. In case no top-down template is suitable after the repeative search, F_2 will be inactive. The learning algorithm of the ART net is summarized in the next paragraph.

ART Learning Algorithm

Step 1. Initialization
Step 2. Apply New Input
Step 3. Compute $U=(f(x_{M+1}), f(x_{M+2}), \ldots, f(x_N))$
Step 4. Select Maximum Exemplar, f(x).

This is performed using extensive lateral inhibition.

Step 5. Vigilance Test

$$\left\{ \frac{|X^*|}{|X|} > \rho \right\} = \begin{cases} NO & GOTO\ STEP\ 6 \\ YES & GOTO\ STEP\ 7 \end{cases}$$

Step 6. Disable Best Matching Exemplar

The output of the best matching node selected in Step 4 is temporarily set to zero, STM RESET WAVE, and no longer takes part in the maximization of Step 4. Then goto Step 3.

Step 7. Adapt Best Maching Exemplar

(2.28) $$z_{ji}(t+1) - z_{ji}(t)\,h(x_i)$$

(2.29) $$z_{ij}(t+1) - \frac{z_{ji}(t)\,h(x_i)}{0.5 + \sum z_{ji}(t)\,h(x_i)}$$

The actual learning of ART is more complicated. It is expressed in differential equation form (Carpenter and Grossberg 1987a). However, the Weber Law Rule and Associative Decay Rule show the form of the equilibrium value of the weight in each learning step. For the choice of the constant values can refer to Carpenter and Grossberg (1987a).

Step 8. Enable any nodes disabled in Step 6. Go to Step 2

Although the mechanism of ART is in advance compared with other unsupervised learning algorithm. Throughout understanding on the ART is not that easy. So that, ART is not being selected as the model in the final year project.

CHAPTER 3: LAR - AN UNSUPERVISED LEARNING ALGORITHM

INTRODUCTION

In this chapter, an unsupervised learning algorithm, the Learning by Attraction and Repulsion (LAR), will be presented. The idea leading to the development of this algorithm for the neural network is inspired from electrostatics phenomenon. In the model of this neural network, we treat the normalized pattern vectors as the position vectors of positive charges, while the normalized weight vectors as the position vectors of negative charges. A step of movement of a negative charge indicates the change of the corresponding weight vector. Hence, according to the Inverse Square Law, $F=k/r^2$, where r is the distance between two charges, we can evaluate each of the forces acting on a single charge as $\Delta W=\mu F$. It is found that the neural network developed from the above idea can self organize to give various responses to different input patterns. Thus, compared with the competitive learning algorithm, this algorithm does not suffer from the limitation due to different initial settings. A simple example in the application of this neural network to pattern recognition is provided.

The discussion will be started from the Initial Setting Problem of the competitive learning. Next, a criterion for pattern recognition by unsupervised neural networks is suggested.

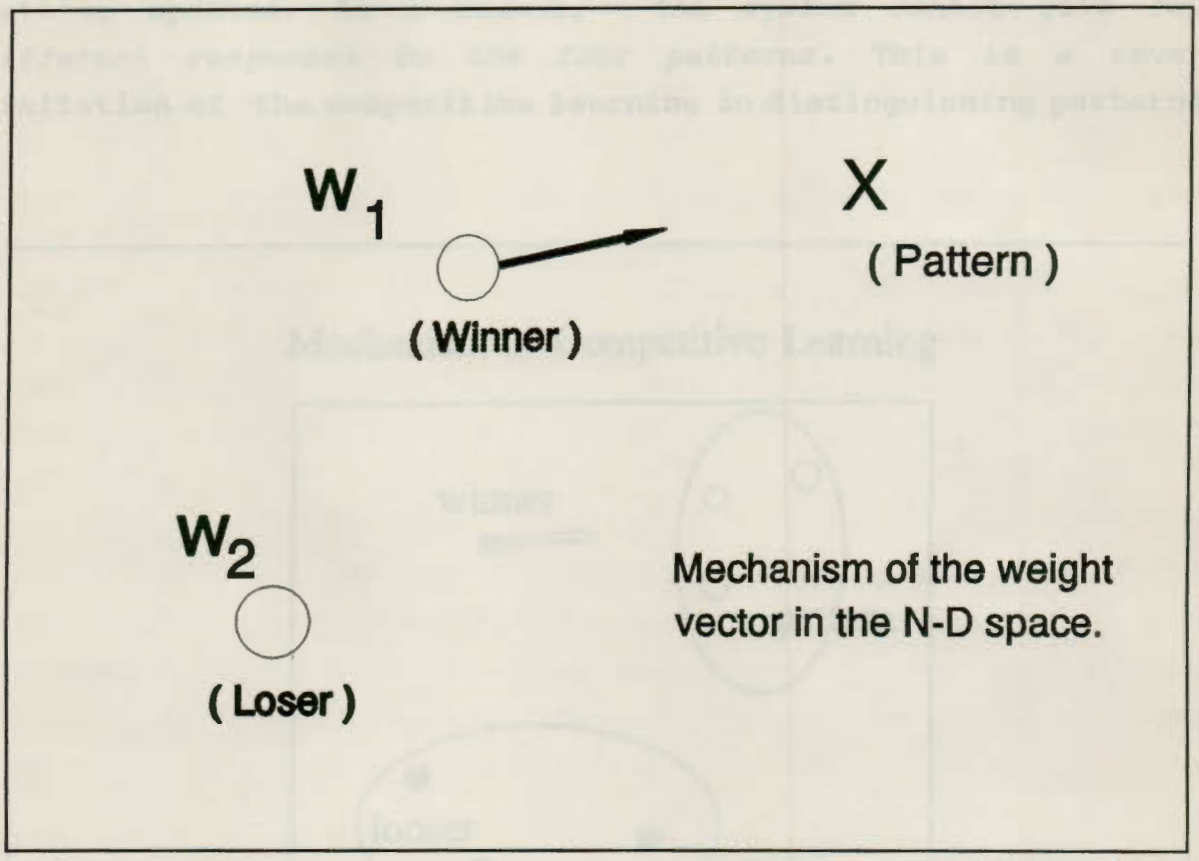
INITIAL SETTING CONSTRAINT

The basic principle of the competitive learning has been presented in chapter 2. Qualitatively, the mechanism can be described as follows.

The winner weight vector will be given to the one which is the

closest to the pattern. Otherwise, it is a loser. For a winner weight, it will be attracted towards the pattern; otherwise, as a loser, it will remain unchange at all.

The step of the movement of the winner is proportional to the distance between the winner weight and the pattern. i.e. $\Delta W=\mu(P-W)$. Diagrammatically, the mechanism can be viewed as Fig(3.1). In any one iteration, only the winner takes the update.

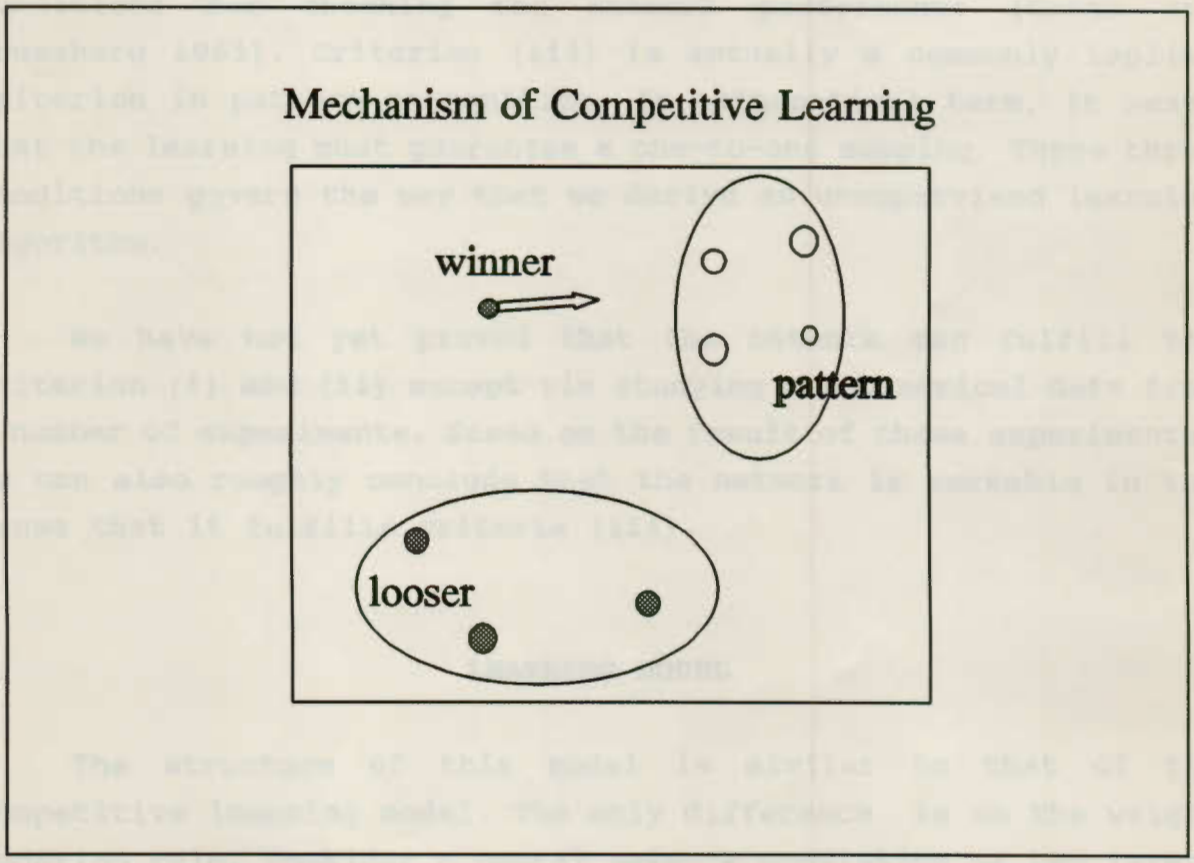


Figure(3.1) Competitive Learning

Under this mechanism, the ability of the competitive learning in pattern recognition is largely dependent on the initial setting, and not just on the similarity amongst patterns. This condition is called the Initial Setting Constraint, (Lee et al 1992a).

Initial Setting Constraint

It is observed that the movement of the neurons is mainly determined by the initial values of both the weight vector and the pattern vector. For instance, in Fig(3.2), we have 4 neurons and 4 input patterns. Initially, only one weight vector is close to all the 4 pattern vectors. It will surely win for all the four patterns. Therefore, it will be the *one and only one* weight vector getting updated. As a result, *the system cannot give four different responses to the four patterns*. This is a severe limitation of the competitive learning in distinguishing patterns.



Figure(3.2) Limitation of Competitive Learning

CRITERIA FOR PATTERN RECOGNITION

The three criteria for building an unsupervised learning network in pattern recognition are:

No Matter which algorithm the neural network is being used, the network must achieve
(i) an equilibrium state, i.e. the final values of the weight vectors must be at equilibrium.
(ii) the equilibrium state must be stable, and
(iii) the output response must be observable.

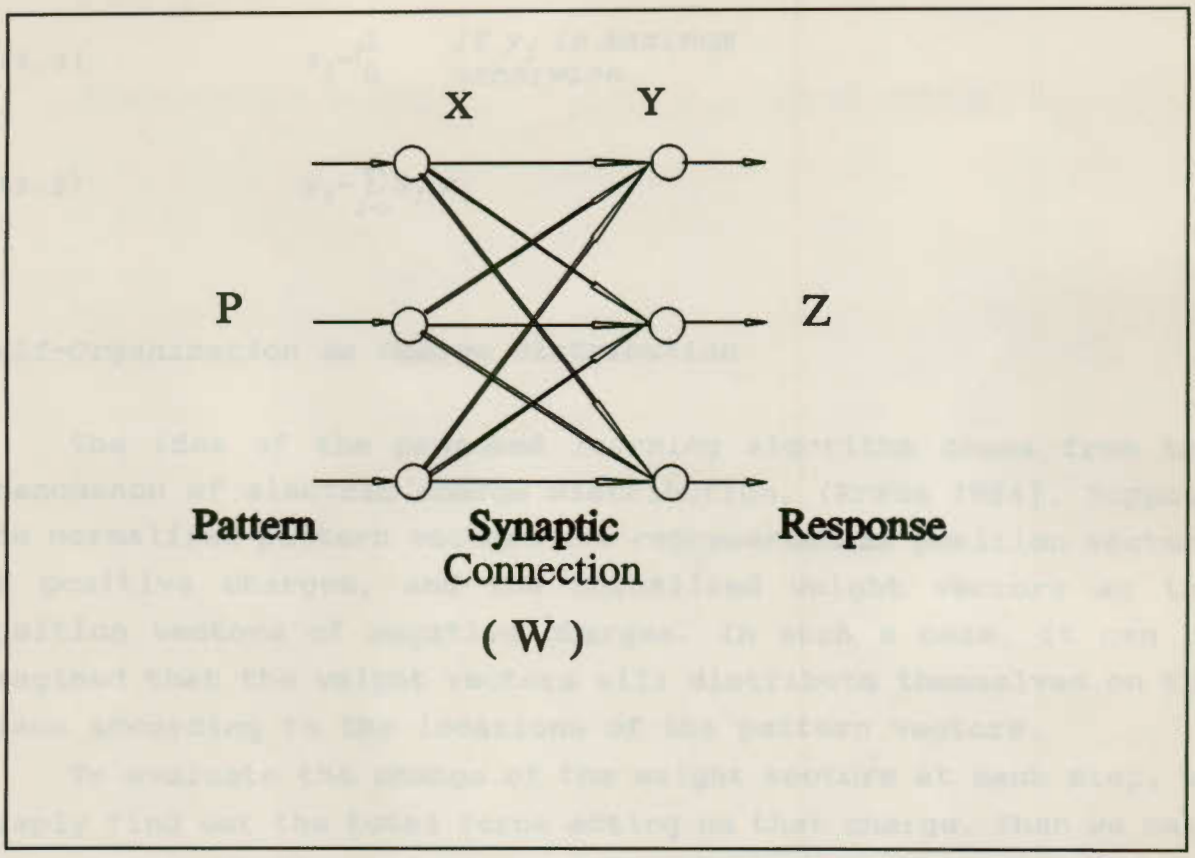
Criteria (i) and (ii), in fact, have been considered as the basic conditions for checking the network performance (Cohen and Grossberg 1983). Criterion (iii) is actually a commonly implied criterion in pattern recognition. In mathematical term, it means that the learning must guarantee a one-to-one mapping. These three conditions govern the way that we derive an unsupervised learning algorithm.

We have not yet proved that the network can fulfill the criterion (i) and (ii) except via studying the numerical data from a number of experiments. Based on the result of these experiments, we can also roughly conclude that the network is workable in the sense that it fulfills criteria (iii).

LEARNING MODEL

The structure of this model is similar to that of the competitive learning model. The only difference is on the weight updating rule. Consider a neural network consisting of two layers of neurons as Fig(3.3).

The first layer contains n neurons while the second layer



Figure(3.3) LAR model

contains m neurons. The organization of this network can be summarized as follows.

Organization

(i) The pattern $\mathbf{P}=(p_1,p_2,\ldots,p_n)$ is impinged onto the first layer of neurons, the responses of the neurons are in *all-or-none* fashion as shown in Equ(3.1) below.

$$x_i=\begin{cases}1 & \text{if } p_i\geq\theta_i \\ 0 & \text{otherwise}\end{cases} \qquad (3.1)$$

(3.2) $$z_j = \begin{cases} 1 & \text{if } y_j \text{ is maximum} \\ 0 & \text{otherwise} \end{cases}$$

(3.3) $$y_j = \sum_{i=1}^{n} w_{ji} x_i$$

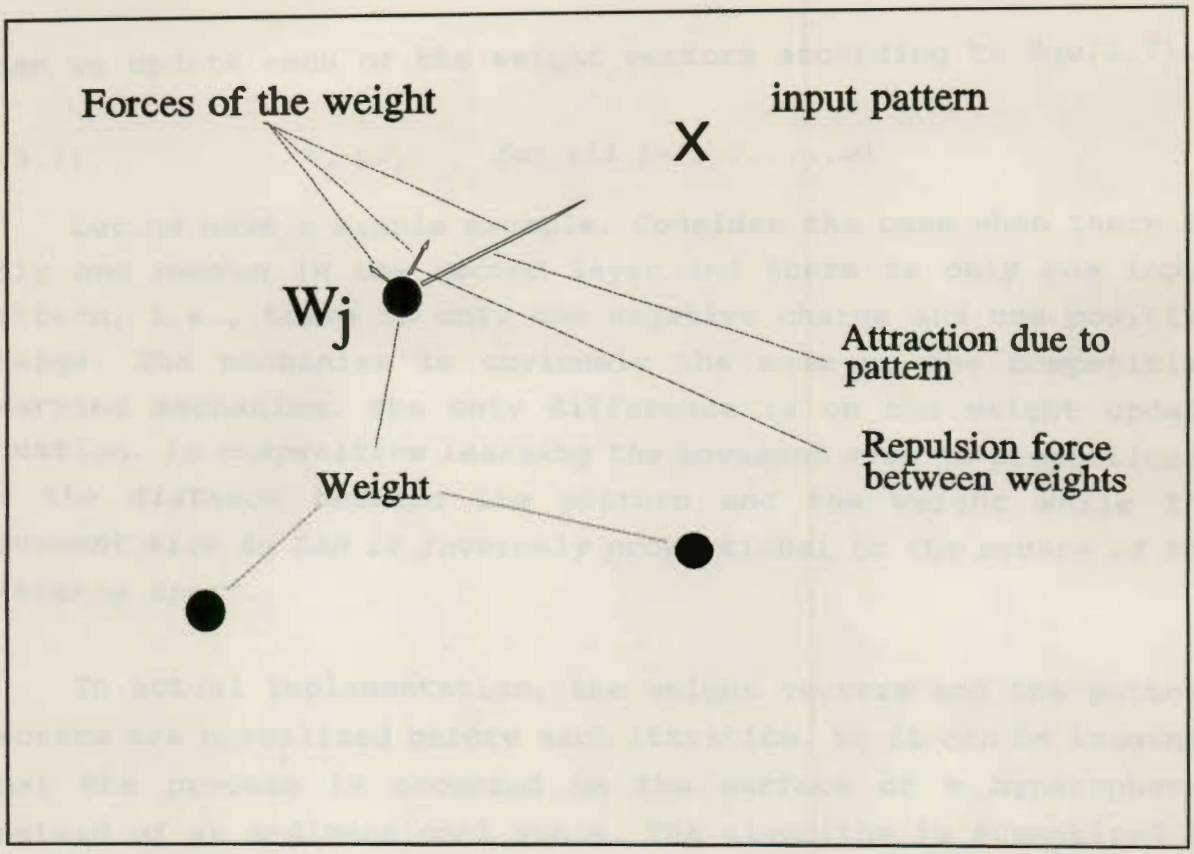
Self-Organization As Charge Distribution

The idea of the proposed learning algorithm comes from the phenomenon of electric charge distribution, (Kraus 1984). Suppose the normalized pattern vectors are represented as position vectors of positive charges, and the normalized weight vectors as the position vectors of negative charges. In such a case, it can be imagined that the weight vectors will distribute themselves on the space according to the locations of the pattern vectors.

To evaluate the change of the weight vectors at each step, we simply find out the total force acting on that charge. Then we make the change of the weight vectors along the direction of the resultant force, i.e. $\Delta \mathbf{W} \alpha \mathbf{F}$. Figure(3.4) shows the above idea.

In Figure(3.4), there are three negative charges in the space with positions given by $\mathbf{W}_j$ where j=1,2,3. There is a positive charge, represented by the cross sign, the position of which is given by the pattern vector. Consider the neuron j. There are three external forces acting on it, indicated by three arrows. According to the Inverse Square Law, the attraction force is given by Equ(3.4).

(3.4) $$F_A = q_1 \frac{X - W_j}{|X - W_j|^3} \qquad \text{where } q_1 = \frac{Q_1 Q_2}{4\pi\epsilon}$$



Figure(3.4) Force diagram

(3.4) $$F_A = q_1 \frac{X - W_j}{|X - W_j|^3} \qquad where\ q_1 = \frac{Q_1 Q_2}{4\pi\epsilon}$$

respectively. Similarly, the repulsion force acting on the neuron j is given by Equ(3.5).

(3.5) $$F_R = q_2 \sum_{r \neq j} \frac{W_j - W_r}{|W_j - W_r|^3} \qquad where\ q_2 = \frac{Q_2 Q_2}{4\pi\epsilon}$$

Hence the resultant force is simply the addition of the attraction and repulsion forces, Equ(3.6).

(3.6) $$F_j = q_1 \frac{X - W_j}{|X - W_j|^3} + q_2 \sum_{r \neq j} \frac{W_j - W_r}{|W_j - W_r|^3}$$

Then we update each of the weight vectors according to Equ(3.7).

(3.7) $\dot{W}_j = \mu F_j$ *for all* $j = (1,2,\ldots,m)$

Let us have a simple example. Consider the case when there is only one neuron in the second layer and there is only one input pattern, i.e., there is only one negative charge and one positive charge. The mechanism is obviously the same as the competitive learning mechanism. The only difference is on the weight update equation. *In competitive learning the movement size is proportional to the distance between the pattern and the weight while the movement size in LAR is inversely proportional to the square of the distance apart.*

In actual implementation, the weight vectors and the pattern vectors are normalized before each iteration. So it can be imagined that the process is occurred on the surface of a hypersphere, instead of an n-dimensional space. The algorithm is summarized as follow.

Step 1. *Normalize all weight vectors to unit magnitude. i.e.*

(3.8) $$\hat{W}_j = \frac{W_j}{\|W_j\|} \qquad \forall j \in \{1,2,\ldots,m\}$$

Step 2. *Select randomly one of the pattern vector. i.e.*

(3.9) $P = P_k$ *where* $k \in \{1,2,\ldots,l\}$

then,

(3.10) $X = \vec{f}(P)$, *also be written as* $x_i = f_i(p_i)$

where,

$$f_i(p_i) = \begin{cases} 1 & \text{if } p_i > 0 \\ 0 & \text{if } p_i \le 0 \end{cases} \tag{3.11}$$

Step 3. *Normalize vector X. i.e.*

$$\hat{X} = \frac{\vec{f}(P)}{\|\vec{f}(P)\|} + \delta \hat{e}_n \qquad \text{where } \delta \to 0 \tag{3.12}$$

$$\hat{e}_n = \left(\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}, \ldots, \frac{1}{\sqrt{n}}\right) \tag{3.13}$$

Step 4. *Calculate the value* F_j, $\forall j \in \{1,2,\ldots,m\}$.

$$F_j = c_1 \frac{\hat{X} - \hat{W}}{\|\hat{X} - \hat{W}\|^3} + c_2 \sum_{r \neq j} \frac{\hat{W}_j - \hat{W}_r}{\|\hat{W}_j - \hat{W}_r\|^3} \tag{3.14}$$

Step 5. *Set the changes of the weight proportional to* F_j. *i.e.*

$$\Delta W_j = \gamma F_j \qquad \forall j \in \{1,2,\ldots,m\} \tag{3.15}$$

and then the new weight vector W_j^* *is set to be*

$$W_j^* = \hat{W}_j + \gamma F_j \tag{3.16}$$

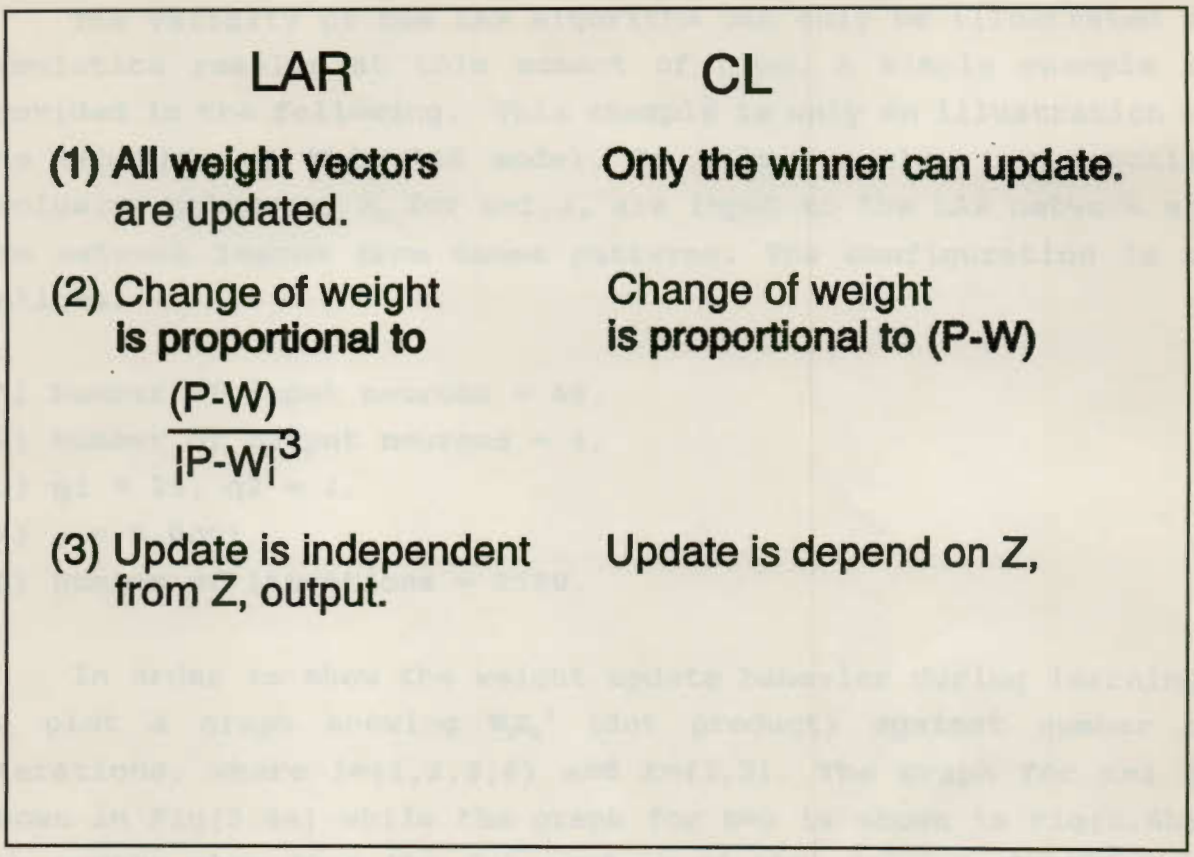
Then set W_j *to equal to* W_j^*, *and goto Step 1.*

The inclusion of Step 3 is to prevent the situation when $\mathbf{W_j = X}$. *In Steps 4 and 5, it is clear that the update of the weight vector is independent of zj, which is the output of the second layer neuron.*

LAR Vs Competitive Learning

Actually, there are at least three differences between the LAR algorithm and the Competitive Learning algorithm, including (i) the

number of weights which can be updated, (ii) the size of movement and (iii) the dependency of the output signal. They are indicated in Fig(3.5).



Figure(3.5) Comparison between LAR and competitve learning

Criteria Fulfillment

As mentioned early in this chapter, a vigorous ***mathematical proof on the three criteria have not yet been established.*** We have so far only experimental results to support our conjecture that the criteria can be fulfilled. In fact, a proof of this algorithm satisfying the criteria has been tried with the Lyapunov method. Lyapunov method is a traditional approach in proving the stability of a number of neural networks (Hopfield 1982, Cohen and Grossberg 1983; Rumelhart and Zipser 1985; Kosko 1992). However, the proof for the LAR model is too difficult to be completed within one

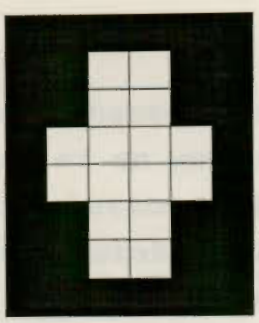
academic year. The difficulty is due to the situation that the learning equation involves the term $(\mathbf{P}-\mathbf{W})/|\mathbf{P}-\mathbf{W}|^3$, instead of $(P-W)$.

The validity of the LAR algorithm can only be illustrated by simulation results at this moment of time. A simple example is provided in the following. This example is only an illustration of the behavior of this LAR model. In this example, two mutually exclusive patterns, X_k for k=1,2, are input to the LAR network and the network learns from these patterns. The configuration is as follows.

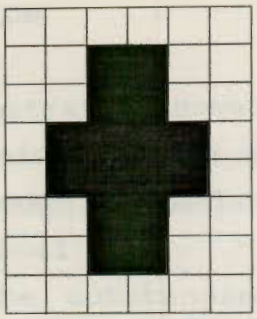
(1) Number of input neurons = 48,
(2) Number of output neurons = 4,
(3) q1 = 25, q2 = 1,
(4) γ = 0.0001,
(5) number of iterations = 2580.

In order to show the weight update behavior during learning, we plot a graph showing $\mathbf{W}_i\mathbf{X}_k^T$ (dot product) against number of iterations, where i=(1,2,3,4) and k=(1,2). The graph for k=1 is shown in Fig(3.6a) while the graph for k=2 is shown in Fig(3.6b). The graphs show that the dot product of the winner weight to the pattern is constant after 640 iterations. It is also observed that the dot product of the losers are not constant after 640 iteration. So it can be imagined that this equilibrium is a dynamic equilibrium. That is to say, the winner is circulating on the surface of a N-D sphere instead of being at a fixed position. According to the graph obtained, there is one weight vector moving around the pattern(1) while there are three weight vectors moving around the pattern(2).

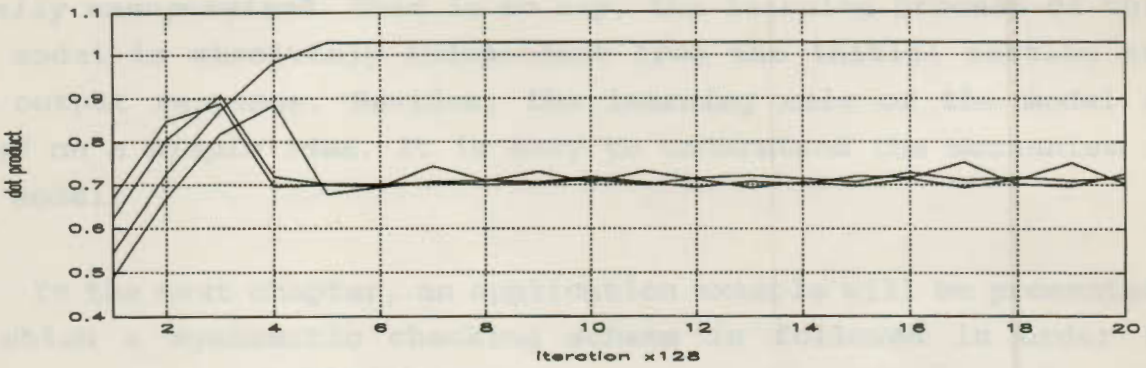
Though the above data cannot be used to prove vigorously that the learning of LAR model is stable for all situations, it does illustrate the validity and behavior of the network.



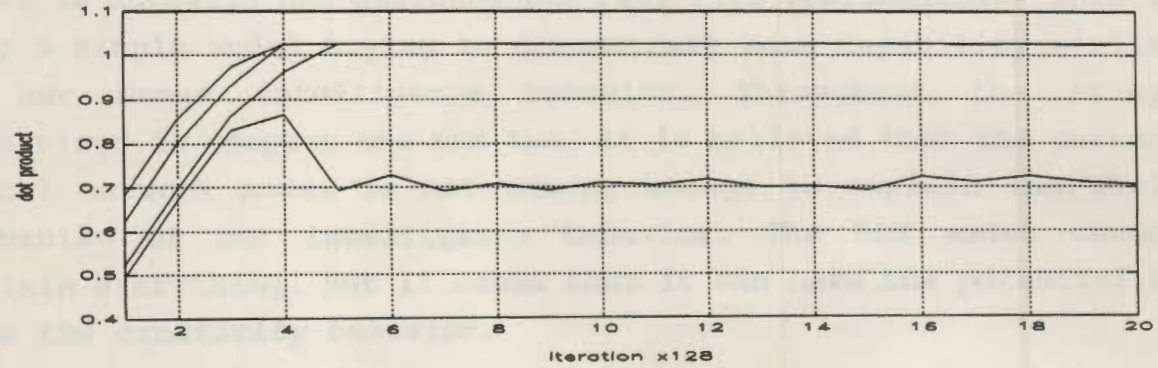
(1)



(2)



(a) Response to (1), $W_i X_1^T$



(b) Response to (2), $W_i X_2^T$

Figure(3.6)

CONCLUSION

In this chapter, we mainly illustrate a novel approach in the development of an unsupervised learning network which is inspired from electrostatics phenomenon. Although the mathematical proof has not yet been established, the numerical data has indicated its ability. The outstanding issue of this model, compared with the traditional neural network, is that ***it is totally unsupervised***. That is to say, the learning process of this LAR model is absolutely independent from the initial setting and the output response. Besides, ***the learning rule of the model is based on a simple idea***. It is easy to understand the mechanism of the model.

In the next chapter, an application example will be presented, in which a systematic checking scheme is followed in order to examine the three criteria.

Here the writer wishes to emphasize several issues. The LAR model is actually not designed for real life applications. This is only a simple model trying to demonstrate some capability similar to our human intelligence behavior. Throughout the study, summarized in chapter one and two, it is believed that the current neural network model is not strong enough to explain the whole mechanism of our intelligence behavior. The LAR model cannot explain everything. But ***it seems that it can have the potential to show the creativity behavior***.

CHAPTER 4 : CHARACTER RECOGNITION USING LAR

INTRODUCTION

In this chapter, we will illustrate an application example of the LAR in character recognition. The following paragraphs will be concentrated on the configuration and the experimental results of the simulation model. Thus, this chapter serves as another experimental example indicating the capability of the LAR.

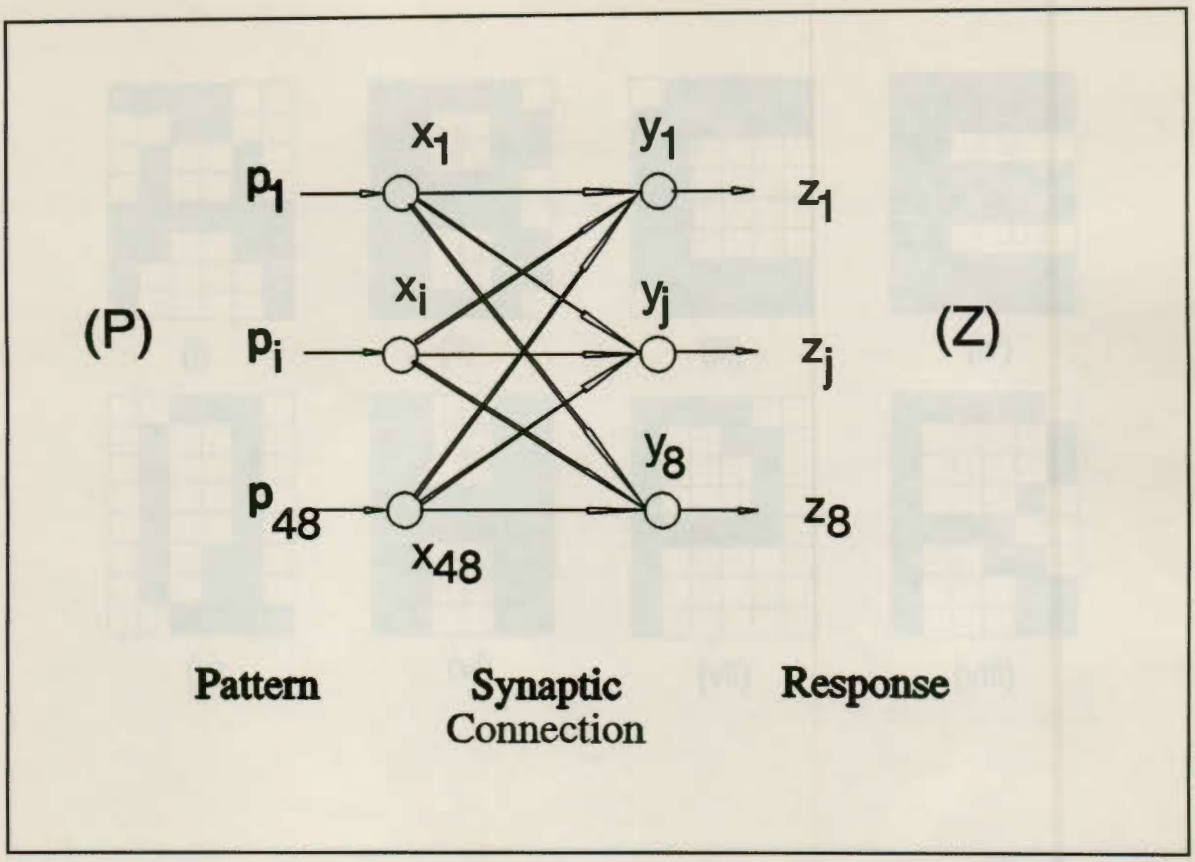
SIMULATION EXAMPLE : Character Recognition

Statement of problem

First of all, let us clarify the aim of this experiment. **One goal of this experiment is to illustrate the ability of the LAR model in differentiating different characters.** That is to say, we wish to show that the network can give different codes for different learnt patterns. The learnt patterns are those characters shown as Fig(4.2). Besides, **another goal of this experiment is to illustrate the incapability of competitive learning in applications to character recognition.**

Structure of the network

Fig(4.1) shows the structure of the neural network. It consists of 48 neurons in the first layer and 8 neurons in the second layer, Fig(4.1). We set the constants q1 and q2 to be 25 and 1 respectively. The step size, γ, is 0.0001. The threshold, θ_i, of each of the neurons in the first layer are set to 0. The update takes over 4096 iterations.



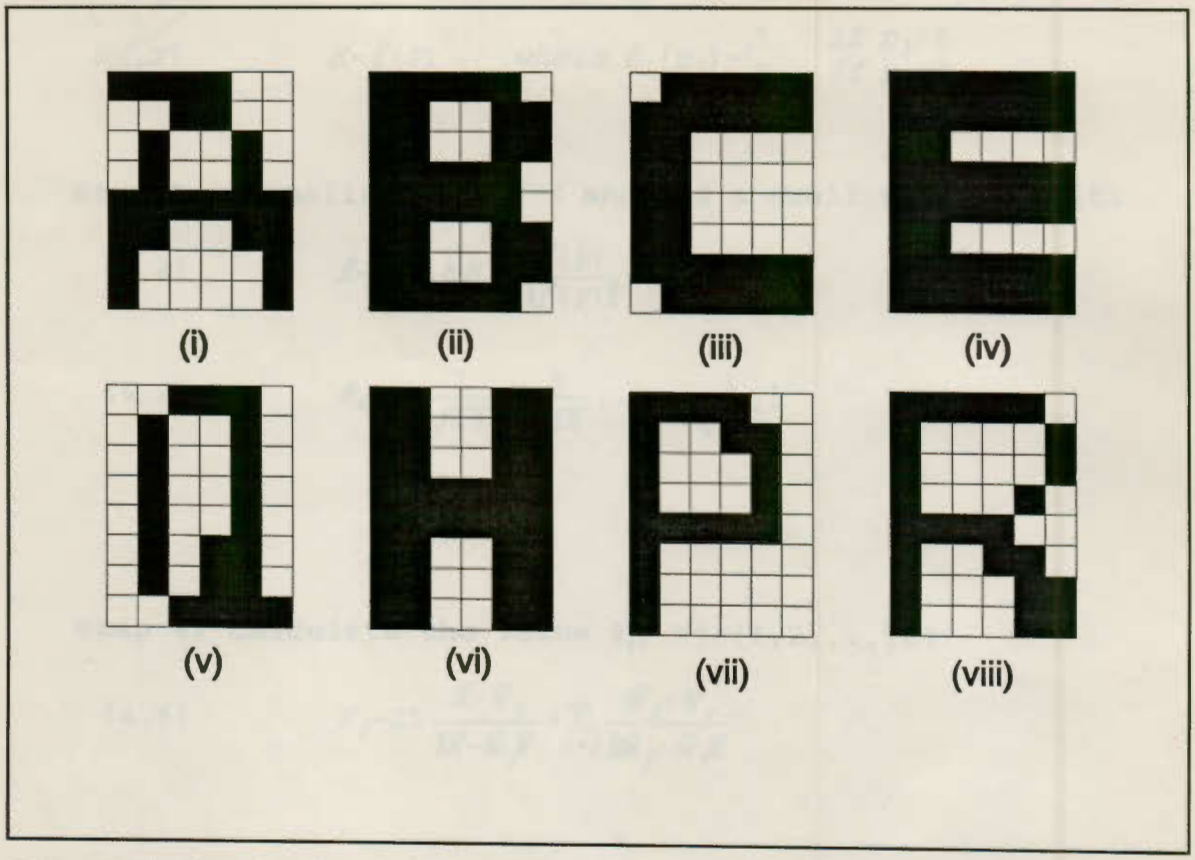
Figure(4.1) Simulation Network

Simulation Result

Two approaches of learning have been applied to the model. One of them is the **general LAR** approach, or called the non-chaotic approach. The simulation result of this approach will be illustrated in subsection Test 1. The other one is the **chaotic LAR** approach and its simulation result is given in subsection Test 2.

Test 1: General LAR approach

In this test, we choose pattern (i) to (vi) in Fig(4.2). The structure of the network is just the same as mentioned in the previous section. The learning algorithm is stated below.



Figure(4.2) Testing characters

Step 1. Normalize all the weight vectors to unit magnitude.

$$\hat{W}_j = \frac{W_j}{\|W_j\|} \qquad \forall j \in \{1,2,\ldots,8\} \tag{4.1}$$

Step 2. Select randomly one of the patterns

$$P = P_k \quad \text{where } k \in \{1,2,3,4,5,6\} \tag{4.2}$$

(4.3)

$$X=\vec{f}(P) \qquad \text{where } f_i(p_i)=\begin{cases}1 & \text{if } p_i>0\\ 0 & \text{if } p_i\le 0\end{cases}$$

Step 3. Normalize vector X and add a small vector to it.

$$\hat{X}=\frac{X}{\|X\|}+\delta\hat{e}_n=\frac{\vec{f}(P)}{\|\vec{f}(P)\|}+0.005\hat{e}_n \tag{4.4}$$

$$\hat{e}_{48}=\left(\frac{1}{\sqrt{48}},\frac{1}{\sqrt{48}},\ldots,\frac{1}{\sqrt{48}}\right) \tag{4.5}$$

Step 4. Calculate the value F_j, $\forall j\in\{1,2,\ldots,8\}$.

$$F_j=25\frac{\hat{X}-\hat{W}_j}{\|\hat{X}-\hat{W}_j\|^3}+\sum_{r\neq j}\frac{\hat{W}_j-\hat{W}_r}{\|\hat{W}_j-\hat{W}_r\|^3} \tag{4.6}$$

Step 5. Set the change of the weight proportional to F_j.

$$W_j^*=\hat{W}_j+0.0001F_j \tag{4.7}$$

Repeat Step 1 to 5 for 4096 iterations.

The simulation results are tabulated below. In case of the LAR model, the 2nd neuron in the second layer is initially the winner of pattern 1 to 4. After 512 iterations, the performance is better. The network can distinguish 4 patterns. After 4096 iterations, the network can completely differentiate all the 6 patterns.

Iteration	Pattern #	zj	max. value of yj

0	1	01000000	0.976141
	2	01000000	0.955053
	3	01000000	1.016840
	4	01000000	1.116477
	5	10000000	1.023682
	6	00000010	1.125428
512	1	00001000	1.159309
	2	01000000	1.157658
	3	00010000	1.146984
	4	10000000	1.253260
	5	00000010	1.194579
	6	00000010	1.290241
4096	1	00001000	1.259999
	2	01000000	1.241957
	3	00010000	1.260759
	4	10000000	1.307193
	5	00000010	1.271767
	6	00000001	1.320562

Stability Checking

Though the results indicate the differentiation of the patterns at around the 4096 iteration, we may query the stability of the response of the network. Can the same response be also achieved if we set the iteration is more than 4096? To show that this can be achieved, we need to observe the variations of all the values of the weights, which is an impossible task. Instead of plotting weight against iterations, y_j is plotted against iterations, $\forall j \epsilon \{1,2,...,8\}$. The value y_j is calculated by Equ.(4.8).

$$y_j = \sum_{i-1}^{48} w_{ji} x_i \qquad (4.8)$$

Fig(4.3) shows the changing of y_j against iteration. After every 128 iterations, the system is tested by the six patterns. For instance, Fig(4.3a) shows the response of the eight output neurons to the character 'A'. The response shows that the network reaches a stable state after 4096 iteration.

Differentiation check

According to the results tabulated. It is found that this network is capable of differentiating different patterns during the 4096 iterations. In addition to the results quoted from the stability checking, we can roughly conclude that the capability of differentiation will be sustained beyond the 4096 iterations. That is to say, the network is workable in the sense of criteria (iii), under this situation.

Test 2: Chaotic LAR approach

In this test, we select the pattern (i) to (iv) and (vii) to (viii) from Fig(4.2). The network parameters are the same. But here we add a concept - **Chaos** - to the simulation program. The mechanism is summarized in the following text.

Step 1. Normalize all the weight vectors to unit magnitude.

$$\hat{W}_j = \frac{W_j}{\|W_j\|} \qquad \forall j \in \{1,2,\ldots,8\} \tag{4.1}$$

Step 2. Select randomly one of the patterns

$$P = P_k \quad \text{where } k \in \{1,2,3,4,5,6\} \tag{4.2}$$

(4.3) $X=\vec{f}(P)$ where $f_i(p_i)=\begin{cases}1 & \text{if } p_i>0\\ 0 & \text{if } p_i\le 0\end{cases}$

Step 3. Normalize vector X and add a small vector to it.

(4.4) $$\hat{X}=\frac{X}{\|X\|}+\delta\hat{e}_n=\frac{\vec{f}(P)}{\|\vec{f}(P)\|}+0.005\hat{e}_n$$

(4.5) $$\hat{e}_{48}=\left(\frac{1}{\sqrt{48}},\frac{1}{\sqrt{48}},\ldots,\frac{1}{\sqrt{48}}\right)$$

Step 4. Calculate the value F_j, $\forall j\in\{1,2,\ldots,8\}$.

(4.6) $$F_j=25\frac{\hat{X}-\hat{W}_j}{\|\hat{X}-\hat{W}_j\|^3}+\sum_{r\neq j}\frac{\hat{W}_j-\hat{W}_r}{\|\hat{W}_j-\hat{W}_r\|^3}$$

Step 5. Set the change of the weight proportional to F_j.

(4.7) $$W_j^*=\hat{W}_j+0.0001F_j$$

After every 128 iterations, select randomly one of the weights and set it to zero.

Repeat Step 1 to 5 for 4096 iteration.

Under this condition, we found that the result is more or less the same as the non-chaotic approach. The y_j is also plotted for reference, indicated in Fig(4.4). In some cases, it is found that the chaotic approach could give a better mode of learning.

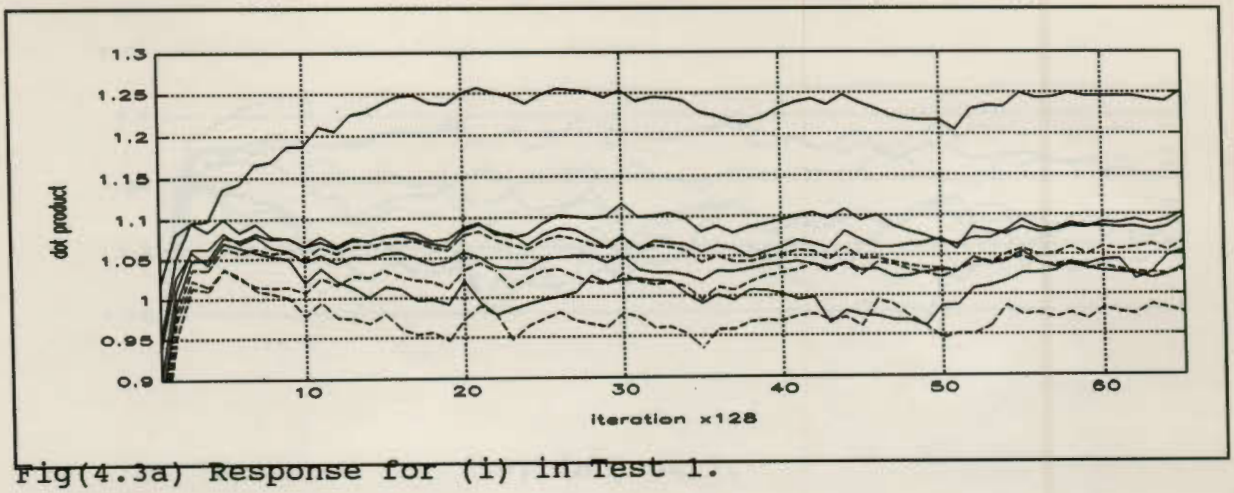
COMPARISON WITH COMPETITIVE LEARNING

The same pattern set is input to the competitive learning

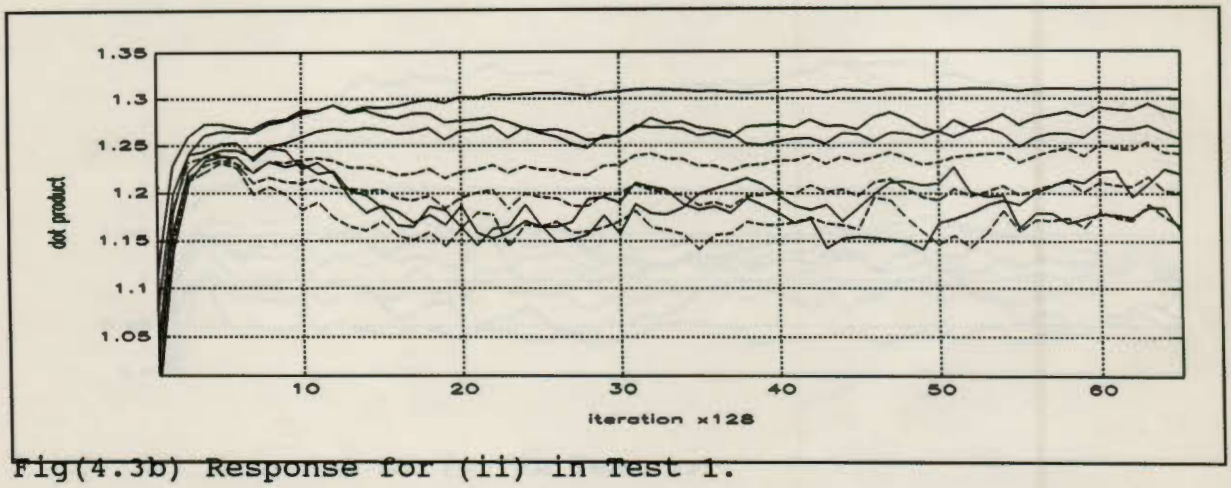
network. The following results are obtained. As expected, the final/intermediate results of the network are totally determined by the initial conditions.

Iteration	Pattern #	zj	max. value of yj
0	1	01000000	0.022365
	2	01000000	0.024384
	3	00010000	0.023920
	4	10000000	0.022417
	5	10000000	0.022807
	6	00000001	0.022316
512	1	01000000	0.022594
	2	01000000	0.024700
	3	00010000	0.024065
	4	10000000	0.022464
	5	10000000	0.022879
	6	00000001	0.022340
4096	1	01000000	0.024251
	2	01000000	0.026696
	3	00010000	0.025121
	4	10000000	0.022746
	5	10000000	0.023439
	6	00000001	0.022494

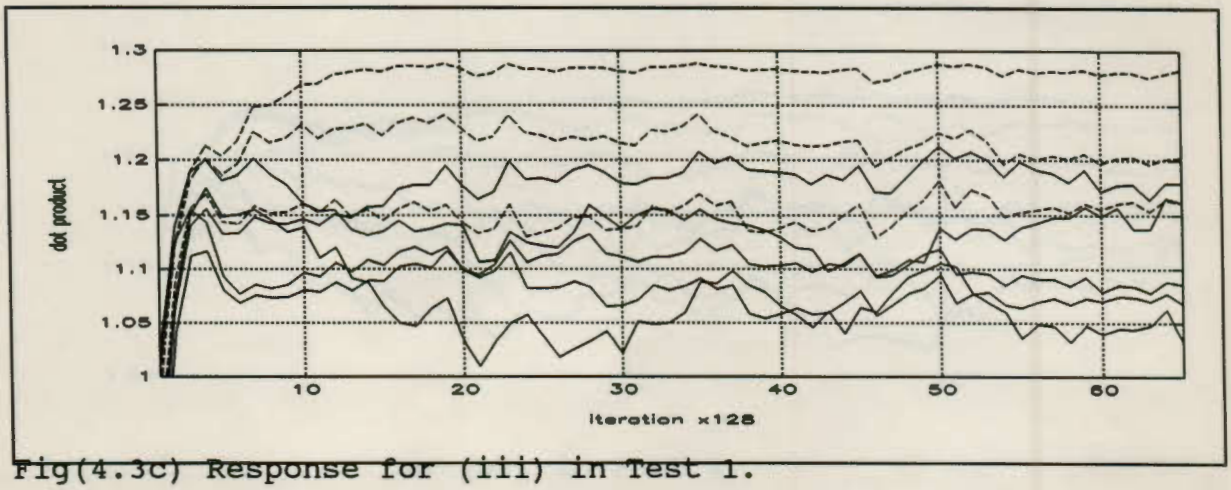
With respect to the differentiation checking, a competitive learning model is found to fail in character recognition.



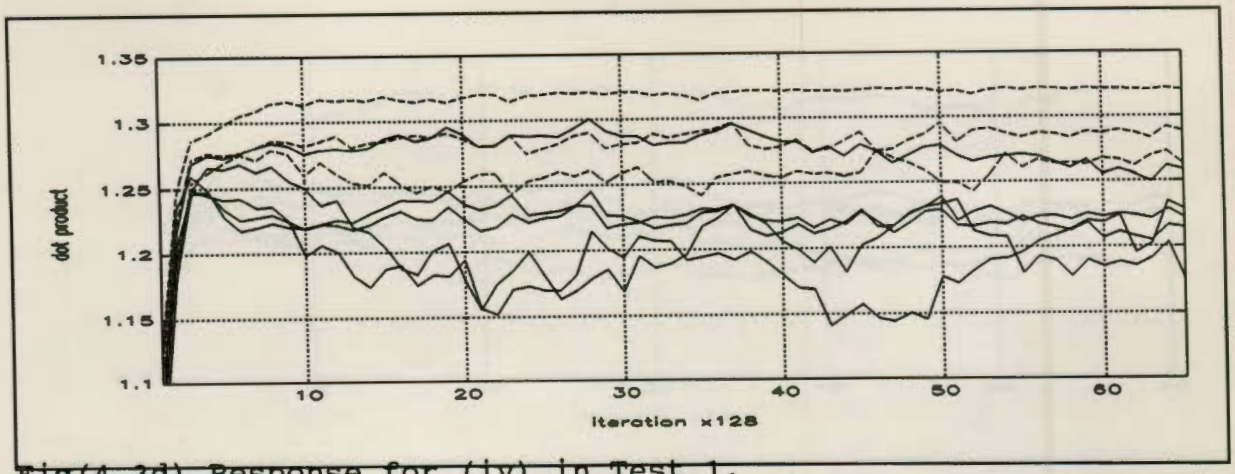
Fig(4.3a) Response for (i) in Test 1.



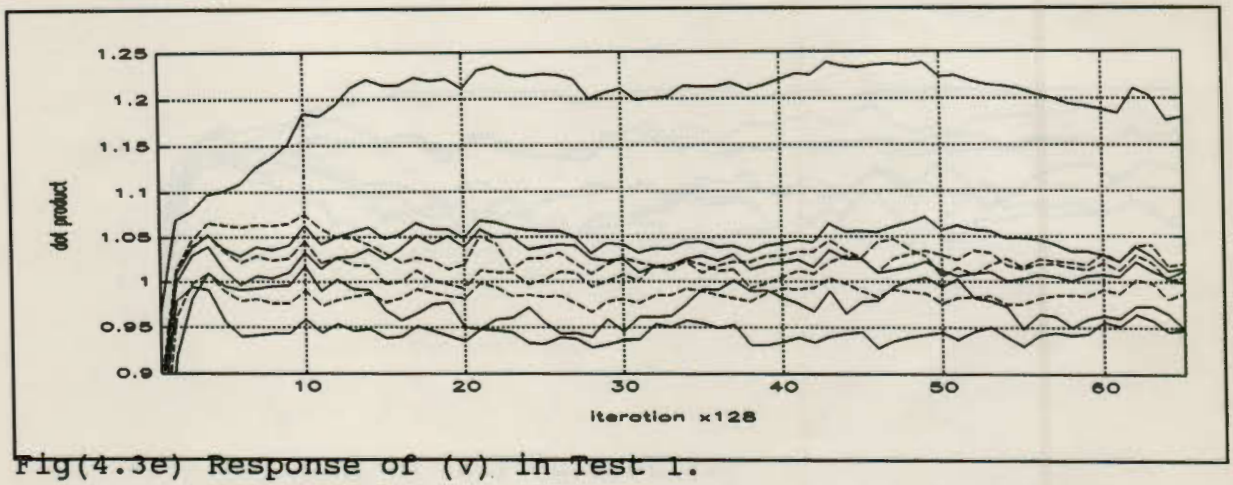
Fig(4.3b) Response for (ii) in Test 1.



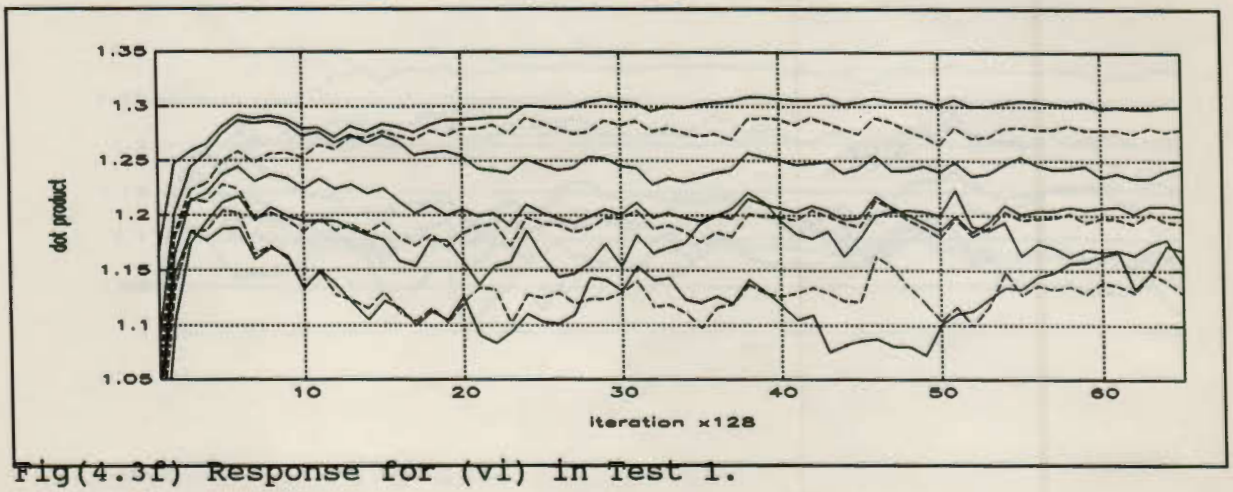
Fig(4.3c) Response for (iii) in Test 1.



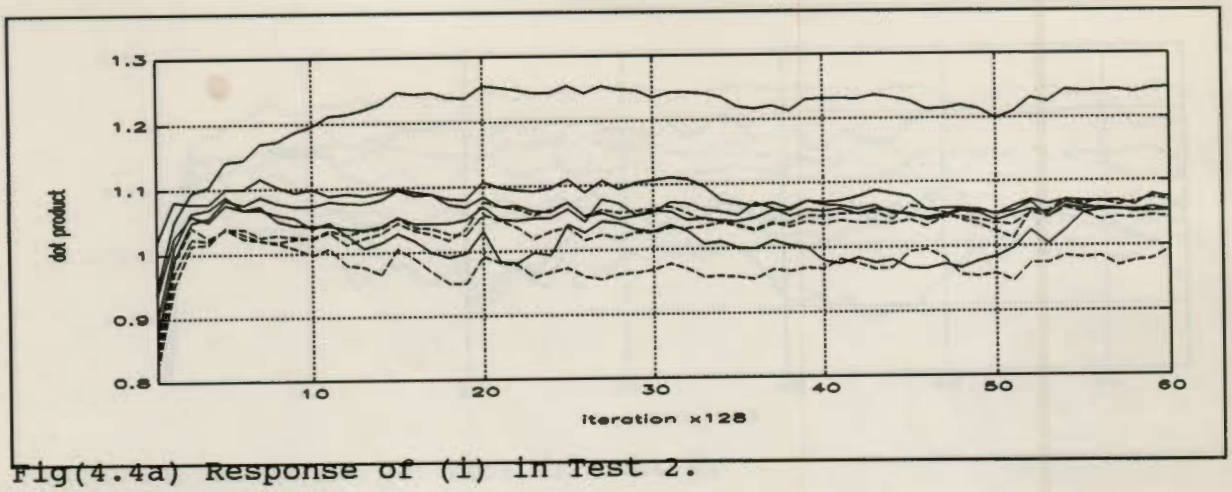
Fig(4.3d) Response for (iv) in Test 1.



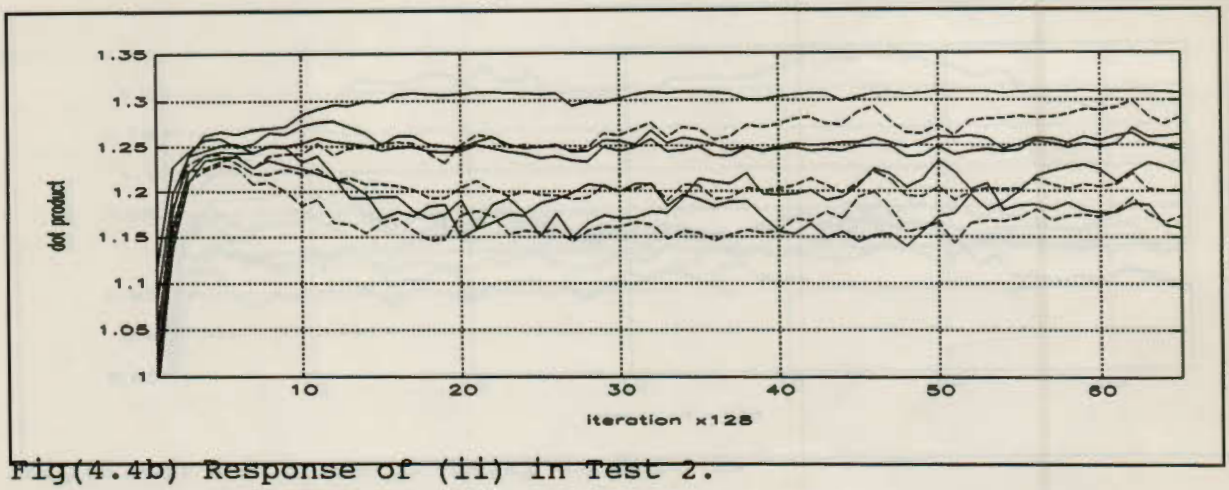
Fig(4.3e) Response of (v) in Test 1.



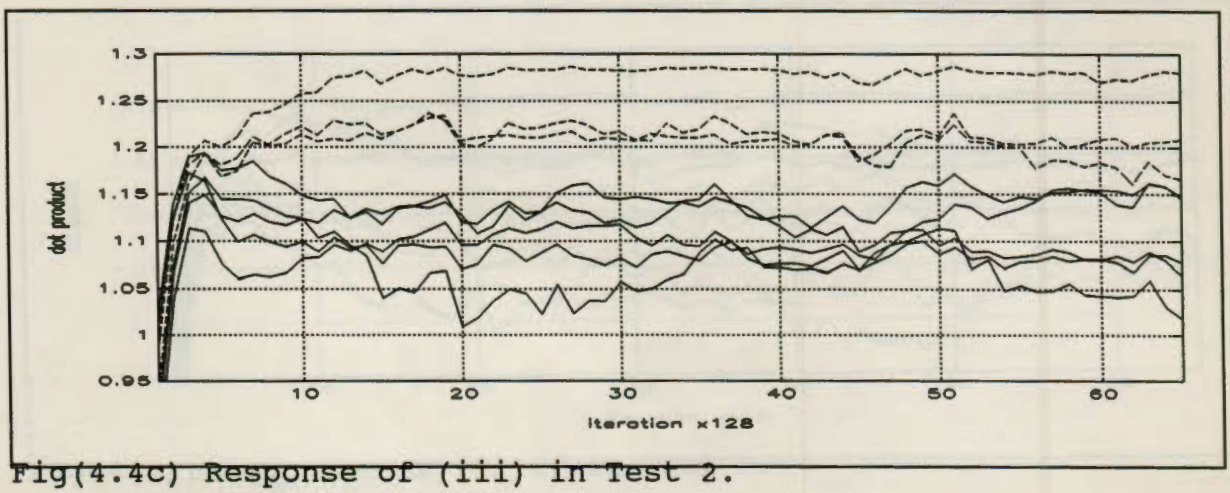
Fig(4.3f) Response for (vi) in Test 1.



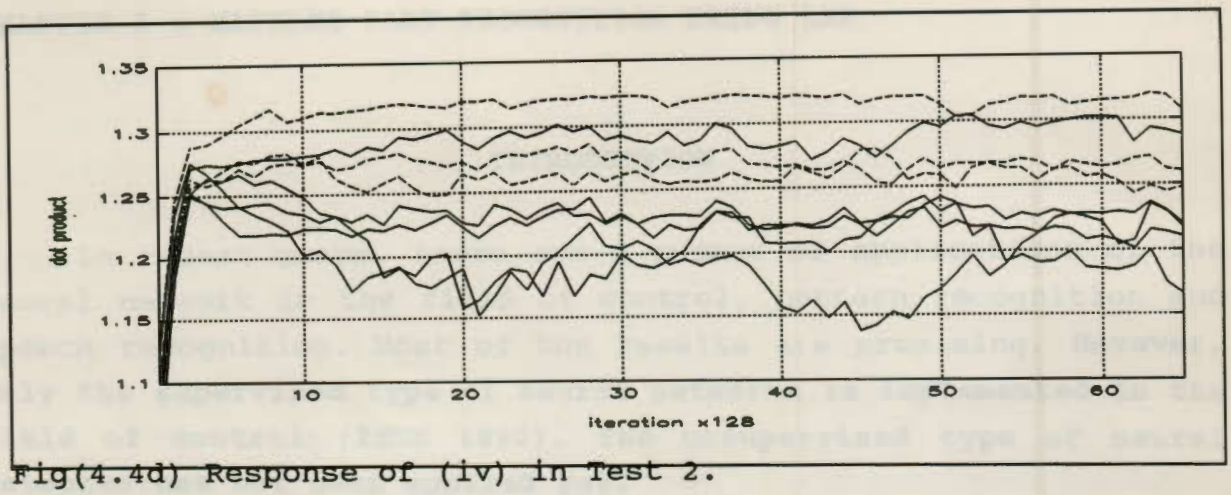
Fig(4.4a) Response of (i) in Test 2.



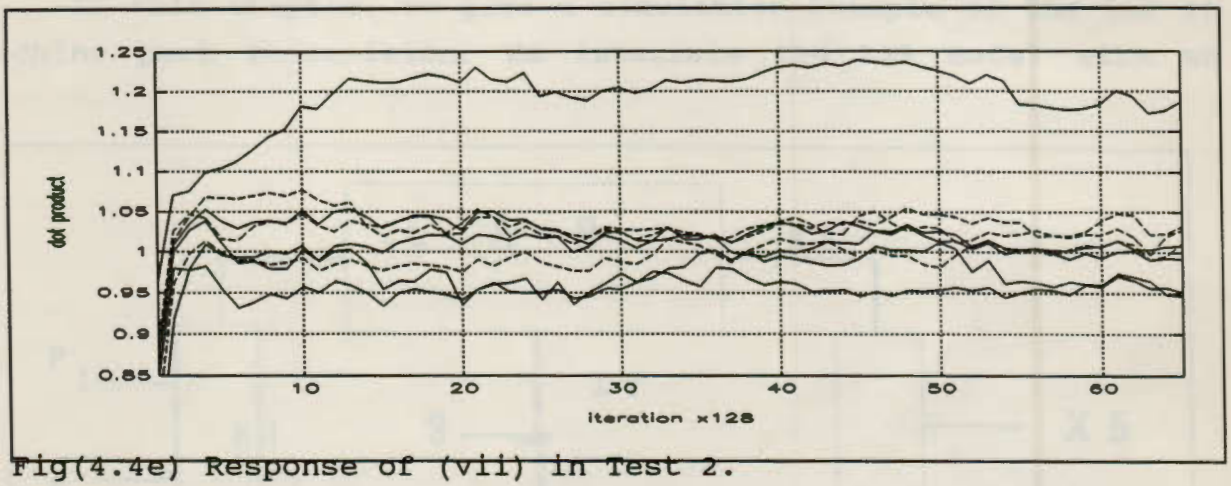
Fig(4.4b) Response of (ii) in Test 2.



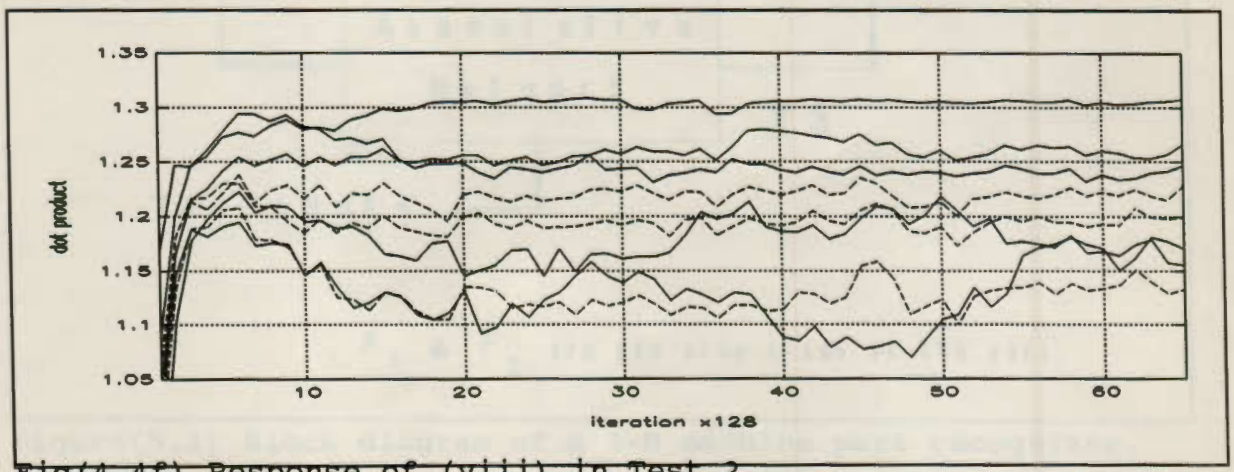
Fig(4.4c) Response of (iii) in Test 2.



Fig(4.4d) Response of (iv) in Test 2.



Fig(4.4e) Response of (vii) in Test 2.



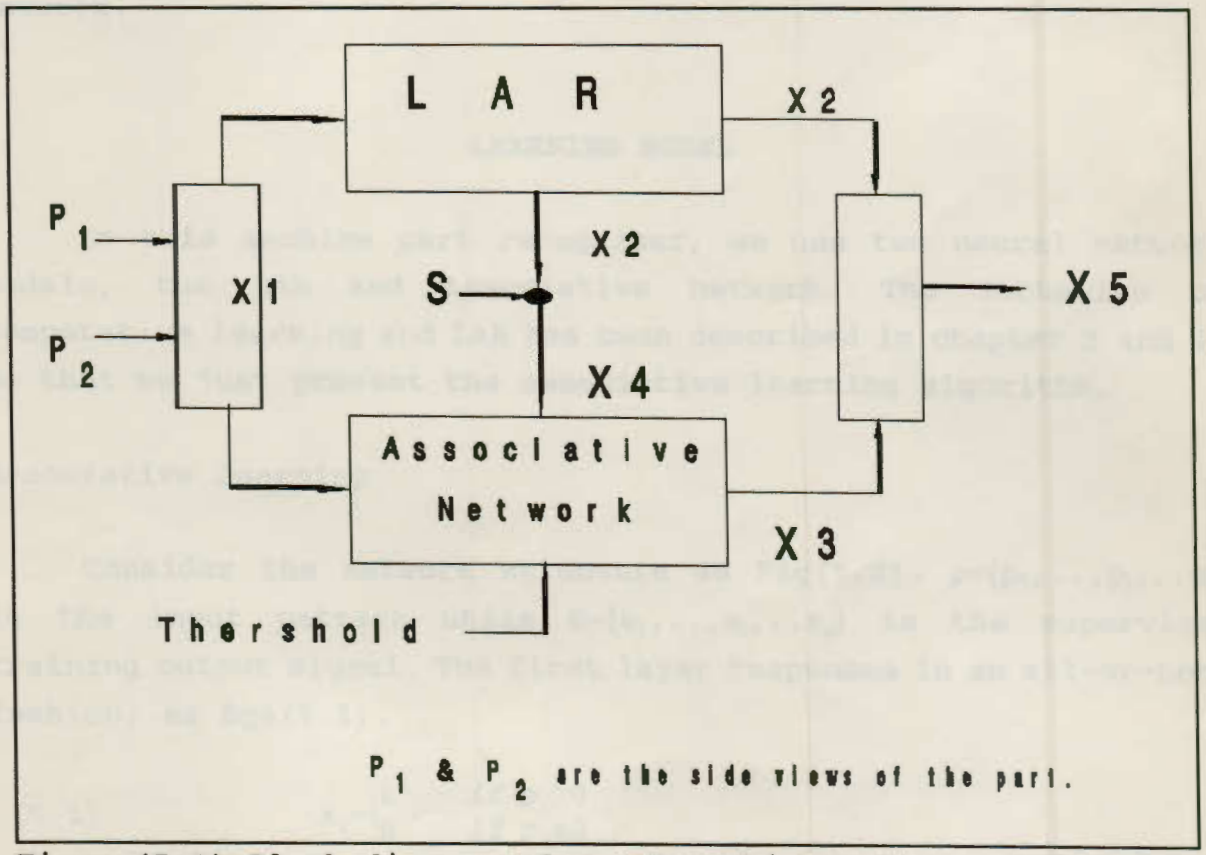
Fig(4.4f) Response of (viii) in Test 2.

CHAPTER 5 : MACHINE PART RECOGNITION USING LAR

INTRODUCTION

In recent years, there are a number of applications of the neural network in the field of control, pattern recognition and speech recognition. Most of the results are promising. **However, only the supervised type of neural networks is implemented in the field of control (IEEE 1992). The unsupervised type of neural networks has not been applied yet.**

In this chapter, we give a simulation example of the LAR in machine part recognition. **We integrate the LAR model with an**



Figure(5.1) Block diagram of a 3-D machine part recognizer.

Associative Learning Model (Anderson 1983). The LAR net classifies the input machine parts into different groups, while the Associative net recognizes the standard defective parts. After Training, this layered neural net can perform the following tasks: *1. to reject the defective machine parts; and 2. to classify the machine parts if they are non-defective*. Apart from 2-D machine part recognition, the same idea can be applied to 3-D machine part recognition, Fig(5.1).

In the next section, we first review on the **learning algorithm**, LAR and Associative net respectively. Then we proceed to elucidate the details on the **organization of the layered neural net**, which is also called the integrated model. **Training procedure** is then given. **Simulation results** are provided for clarification of the organization and the training procedure of the layered neural network.

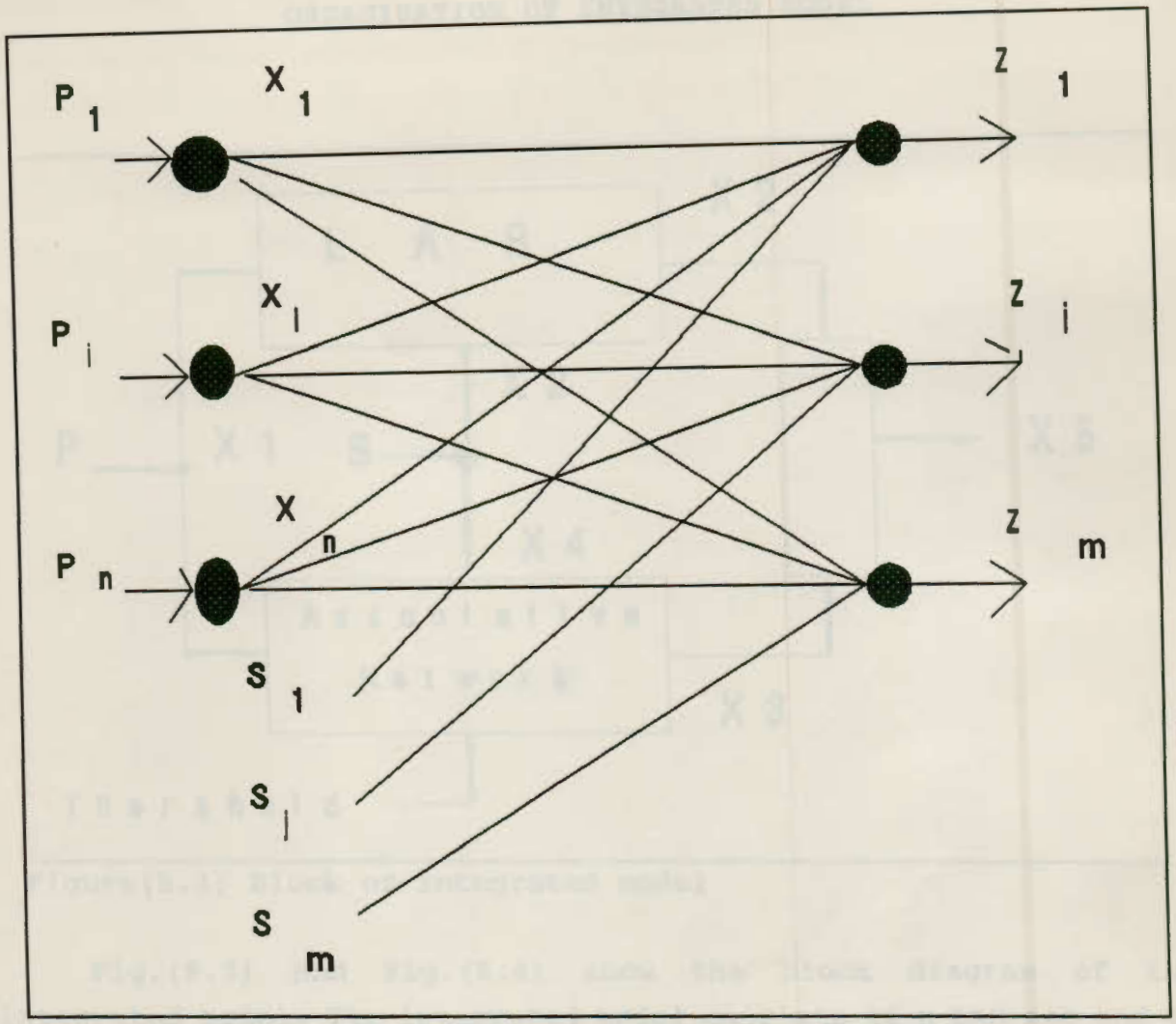
LEARNING MODEL

In this machine part recognizer, we use two neural network models, the LAR and Associative network. The mechanism of competetive learning and LAR has been described in chapter 2 and 3. So that we just present the associative learning algorithm.

Associative Learning

Consider the network structure as Fig(5.2). $\rho=\{p_1,..,p_i,..p_n\}$ is the input pattern while $S=\{s_1,..,s_i,..s_n\}$ is the supervised training output signal. The first layer responses in an all-or-none fashion, as Equ(5.1).

$$x_i=\begin{cases}1 & \text{if } p_i>0\\ 0 & \text{if } p_i\le 0\end{cases} \qquad (5.1)$$



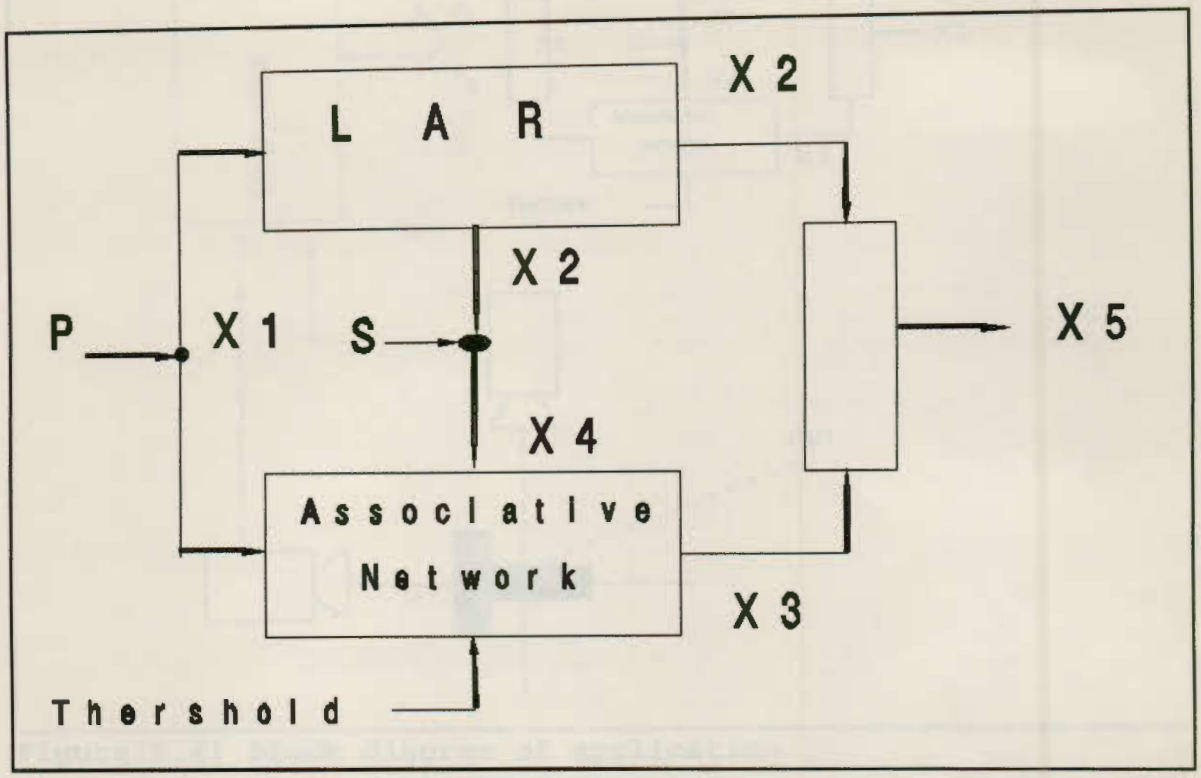
Figure(5.2) Associative learning network

During learning, we input $\{\rho, \mathbf{s}\}$ pairs to the network and let it learn as Equ(5.2).

$$\Delta w_{ji} = -\alpha w_{ji} + \beta s_j x_i \tag{5.2}$$

where α, β are constants. For details of this learning model, refer to Anderson (1983) and Hecht-Nielsen (1990).

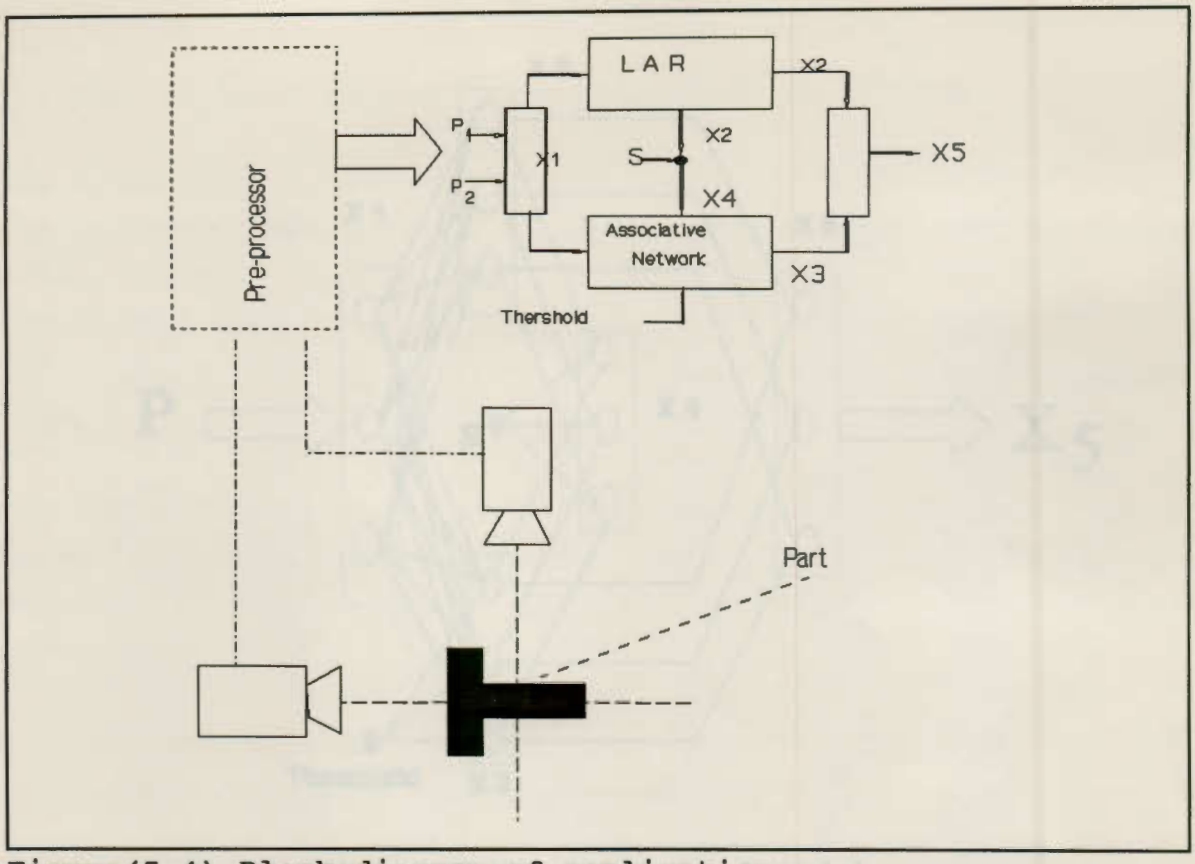
ORGANIZATION OF INTEGRATED MODEL



Figure(5.3) Block of integrated model

Fig.(5.3) and Fig.(5.4) show the block diagram of the integrated model. The integrated model consists of a LAR net and an Associative net. The LAR net is trained to recognize the non-defective machine parts, while the Associative net is being trained to recognize the defective machine parts.

After training of the LAR net and Associative net sequentially, these two networks can cooperate to perform the task - classification the non-defective machine parts and rejection of those defective parts. Fig.(5.5) shows the structure of the integrated model. The nodes in the first layer are partitioned into two groups. The top view (side 1) and the lateral view (side 2) of the machine parts are input respectively. The overall operation



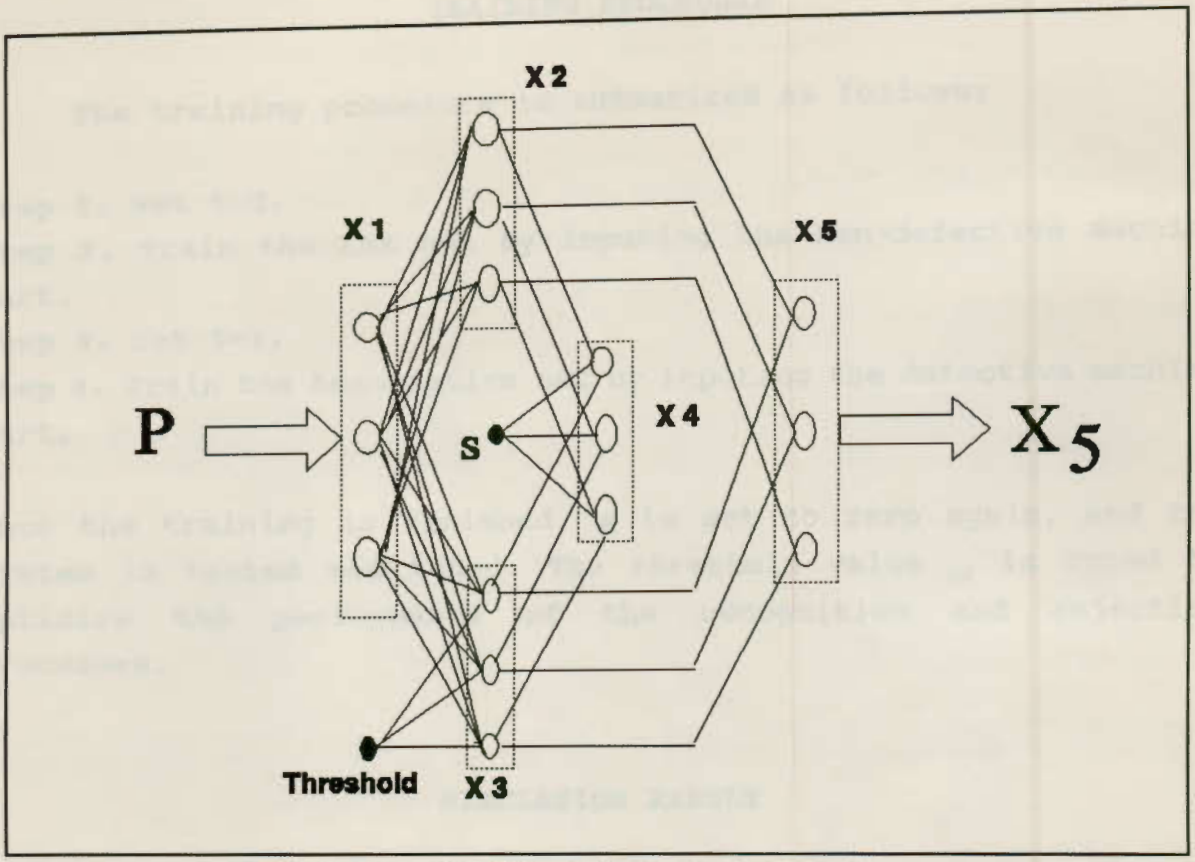
Figure(5.4) Block diagram of application

principle is summarized by Equ(3) to Equ(7).

(5.3) $$x_i(1) = \begin{cases} 1 & \text{if } p_i > 0 \\ 0 & \text{otherwise} \end{cases}$$

(5.4) $$x_i(2) = \begin{cases} 1 & \text{if } \sum_j w_{ij}(1)\, x_j(1) \text{ is max} \\ 0 & \text{otherwise} \end{cases}$$

(5.5) $$x_i(3) = \begin{cases} 1 & \sum_j w_{ij}(2)\, x_j(2) > \theta_i \\ 0 & \text{otherwise} \end{cases}$$



Figure(5.5) Structure of the integrated model

(5.6) $$x_i(4) = \begin{cases} 1 & \text{if } (S \wedge x_i(2)) = 1 \\ 0 & \text{otherwise} \end{cases}$$

(5.7) $$x_i(5) = \begin{cases} 1 & \text{if } (x_i(2) > x_i(3)) \\ 0 & \text{otherwise} \end{cases}$$

where θ_i is the thresholds values of the neurons on layer X(3) and s is the supervisor signal, given by Equ(5.8), provided only in the training mode.

(5.8) $$s = \begin{cases} 1 & \text{if the part is defective} \\ 0 & \text{if the part is non-defective} \end{cases}$$

TRAINING PROCEDURE

The training procedure is summarized as follows:

Step 1. Set S=0.
Step 2. Train the LAR net by inputing the non-defective machine part.
Step 3. Set S=1.
Step 4. Train the Associative net by inputing the defective machine part.

Once the training is finished, s is set to zero again, and the system is tested and tuned. The threshold value e_j is tuned to optimize the performance of the recognition and rejection processes.

SIMULATION RESULT

A simulation program has been written for illustrating the algorithm. The parameters of the structure are listed below.

(a) number of neurons in X(1) = 72[1]
number of neurons in X(2) = 4
number of neurons in X(3) = 4
number of neurons in X(4) = 4
number of neurons in X(5) = 4

(b) LAR net : q_1=25, q_2=1 ;
γ = 0.0001,
number of iteration = 7680.

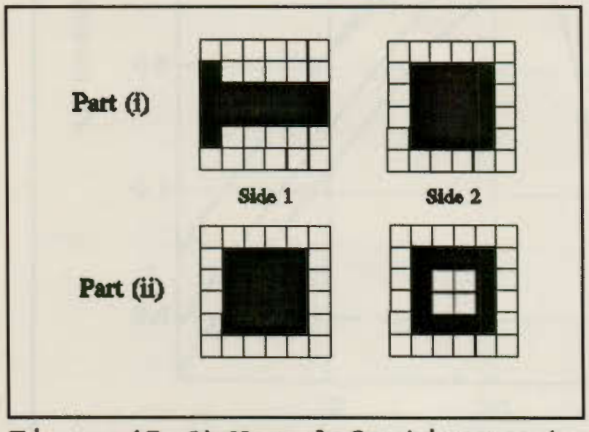
[1] 36 neurons are for one side of view.

(c) Associative net : Θ_j = 0.95, 0.975, 1.

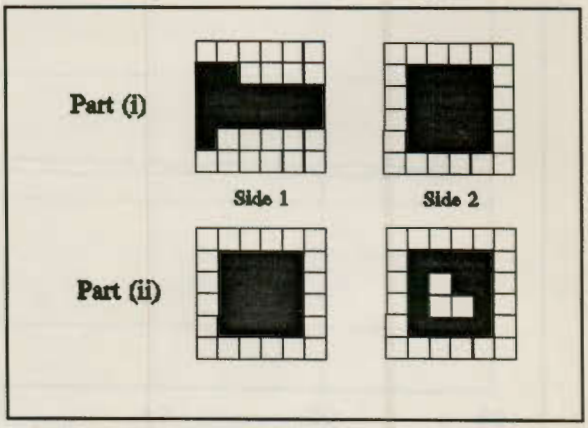
α = 0.01, β = 0.05 and

number of iteration = 7680

One set of machine parts are being tested[2], Fig.(5.6). While the learning of the non-defective part is finished, the



Figure(5.6) Non-defective parts



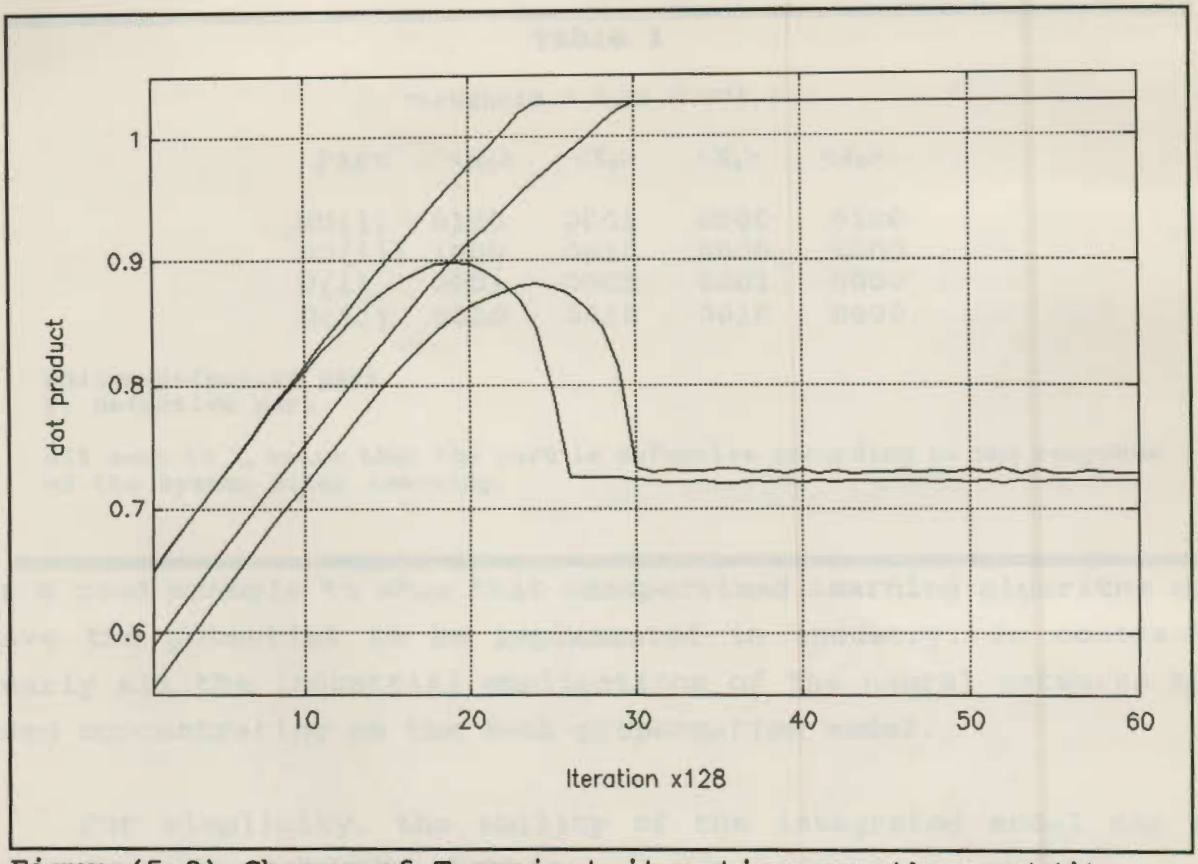
Figure(5.7) Standard defective parts

system is trained with a set of defective machine parts, Fig.(5.7). In order to test the ability of the system in differentiating patterns, the two defective parts are designed to be very similar to the non-defective part, only different from one pixel. These machine parts are then treated as the standard and used to train the associative network.

While the LAR is being trained, the values $T_i = \sum w_{ij}(1)x_j(1)$, the dot product, is recorded. There are four curves. Each of them representing the response of one neuron in the second layer. These curves only indicate the change of T_i for the part(i). The trend of the changing of T_i is shown in Fig.(5.8). This plot indicates that

---

[2] Each side-view of the machine parts are represented by binary signal. For example the side 1 of the set(1) is in the form of {00000001111001111001111001111000000}.



Figure(5.8) Change of T against iteration, on the part(i).

the LAR can achieve a stable state while learning. It is shown that there are two winner neurons, in the X_2 layer, giving the same dot product value, approx. 1.05.

After all the training, the integrated model is tested and the results are tabulated in Table 1. It is found that the system can correctly reject or classify the machine parts for all the threshold settings.

CONCLUSION

Throughout the paper, we are trying to illustrate a simple application example of an unsupervised learning model, the LAR. It

Table 1

Threshold = 0.95,0.975,1.

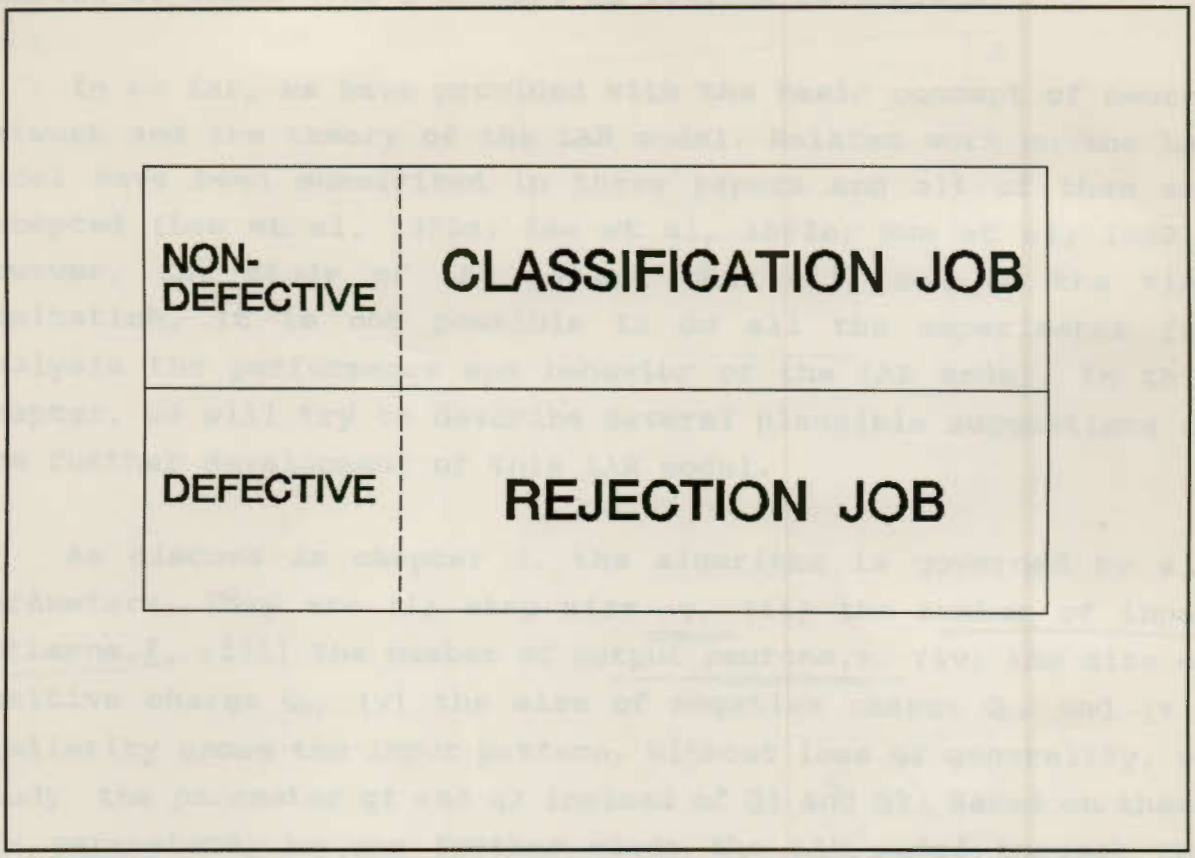
Part	$<X_2>$	$<X_3>$	$<X_4>$	$<X_5>$
ND(i)	0100	0001	0000	0100
ND(ii)	1000	0010	0000	1000
D(i)	0001	0001	0001	0000
D(ii)	0010	0010	0010	0000

ND:non-defective part
D: defective part

All zero in X_5 means that the part is defective according to the response of the system after learning.

is a good example to show that unsupervised learning algorithm can have the potential to be implemented in industry. In contrast, nearly all the industrial applications of the neural networks had been concentrating on the back-proporgation model.

For simplicity, the ability of the integrated model can be viewed as shown in Fig.(5.9).



Figure(5.9) Functions of the integrated model

CHAPTER 6: CONCLUSION & COMMENT ON FURTHER DEVELOPMENT

In so far, we have provided with the basic concept of neural network and the theory of the LAR model. Related work on the LAR model have been summarized in three papers and all of them are accepted (Lee et al, 1992a; Lee et al, 1992b; Sum et al, 1992). However, the study of LAR is not that all. Due to the time limitation, it is not possible to do all the experiments for analysis the performance and behavior of the LAR model. In this chapter, we will try to describe several plausible suggestions on the further development of this LAR model.

As discuss in chapter 3, the algorithm is governed by six parameters. They are (i) step size, γ, (ii) the number of input patterns, l, (iii) the number of output neurons, n, (iv) the size of positive charge Q_1, (v) the size of negative charge Q_2, and (vi) similarity among the input pattern. Without loss of generality, we study the parameter q1 and q2 instead of Q1 and Q2. Based on these six parameters, we can further study the LAR model through the following ways.

(1) Analysis of the performance of the model on step size.

In fact, it just repeats the experiment presented in chapter 4 but with a smaller step size.

(2) Analysis of the influence of l/n.

In this test, all parameters except l are fixed. γ=0.0001, n=16, q_1=25 and q_2=1. Then increment the number of input patterns from 1 to 16. Plot a graph showing the differentiation ability against the ratio l/n.

(3) Analysis of the influence of q_1/q_2.

Also, we can fix all the parameters except q_1. γ=0.0001, n=l=2, q_2=1. Then set the value q_1 changing from 1 to 50, one at a time.

Plot a graph showing the differentiation ability against the ratio q_1/q_2.

(4) Analysis of the pattern similarity on the performance of the model.

Fix the value of parameters as follows: γ=0.0001, n=1=2, q_1=25 and q_2=1. Here the correlation factor is defined as Equ(6.1).

$$C=\frac{P_1 \cap P_2}{|P_1||P_2|} \qquad \text{where } P_k=(p_{k1}, p_{k2}, \ldots, p_{kn}) \tag{6.1}$$

and

$$P_1 \cap P_2=\sum_{i=1}^{n} p_{1i}p_{2i} \qquad \text{where } p_{ki} \in \{0, 1\} \tag{6.2}$$

By inputing different pairs of patterns to the network, a graph can be plotted showing the differentiation ability against C.

(5) Find out a mathematical proof or disproof on whether the LAR model can fulfill the criteria mentioned in chapter 3.

(6) Study the application scope of the LAR model.

6×8

random generate patterns

$l = 2$
3
$l = 4$
5
6

Expl 1: $l = 6$, $\longrightarrow$ $n = 6$
$n = 7$
$n = 8$
$n = 9$
$n = 10$
$n = 11$
$n = 12$
$n = 13$
$n = 14$
$n = 15$
$n = 16$

n_0 is the minimum # of n such that it can totally different pattern.

$n_0 = l$

for 6x9

l

7
$l = 8$
9
10
11
12
13
14
15
16

< *APPENDIX A* >

```
Directory of A:\LAR

CHAR01    DAT        301 12-24-91   2:10a
CHAR02    DAT        400 01-10-92   5:48p
CMCL      C         4464 01-09-92  11:11p
CMCL      OBJ       5853 01-09-92   1:22a
CMCL      EXE      36397 01-09-92   1:22a
MCL       C         4245 01-09-92  11:01p
MCL       OBJ       5651 01-09-92  11:01p
MCL       EXE      36234 01-09-92  11:01p

Directory of A:\MPR

MPR01     C         8300 01-29-92   3:26a
MPRG01    DAT         99 01-28-92   3:31p
MPRD01    DAT         99 01-28-92   3:38p
MPR01     OBJ      10337 01-29-92   3:32a
MPR01     EXE      37942 01-29-92   3:32a
MPR01     BAK       8303 01-29-92   3:24a
MPRD01    BAK         99 01-28-92   3:31p
```

< APPENDIX A(i) >

```
/*-------------------------------------------------------------------------
      CMCL.C      8-JAN-92

      Modified Compeititve Learning with Chaos.

      - The rule of response in the second layer is followed
        the competitive learning.
      - The weight updating rule is a new, not the same as
        the classic one. Its idea is based on a phenomena in
        electrostatics, like pole repell and unlike pole attract.
      - Chaos idea has been implemented in the algorithm to
        show the effect.

-------------------------------------------------------------------------*/

#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<conio.h>
#include<graphics.h>

#define iteration 4096
#define inneuron 48
#define outneuron 8
#define image     6
#define     T     0.0001
#define mass      25


int   X2[outneuron];
float X1[image][inneuron];
float DWW[outneuron][outneuron];
float DWX[outneuron][image];
float IL[image];
float W1[outneuron][inneuron];
float WN[outneuron];

void  chaos(void);
void  forword_respond(void);
void  inputpattern(void);
void  learning(void);
void  loadinput(void);
void  normalization(void);
void  normalize_image(void);
void  testing(void);
void  weight_init(void);
void  weight_norm(void);



main()
{
int   iterate;
int   index;
void  print_weight();
loadinput();
normalize_image();
weight_init();
for(iterate=0; iterate<iteration; iterate++)
```

```
  {
  /*printf(".", iterate);*/
  learning();
  weight_norm();
  normalization();
  index=iterate%256;
  if (index==0) { printf("\n\n --- %d ---\n\n",iterate);
                  print_weight();
                  forword_respond();
                  printf("\n\n");
                  chaos();    }
  }

printf("\n\n --- %d ---\n\n",iterate);
print_weight();
forword_respond();
}


void loadinput()
{
char  xc;
int   i,j,k;
int   xx;
FILE  *input;

/*input=fopen("mcl.dat","r");*/
input=fopen("char01.dat","r");
for(k=0; k<image; k++)
  {  for(i=0; i<inneuron; i++)
     {   xc=getc(input);
       xx=atoi(&xc);
       X1[k][i]=xx;
     }
     getc(input);
  }
fclose(input);
}



void weight_init()
{
int   i,j;

for(i=0; i<outneuron; i++)
  for(j=0; j<inneuron; j++) W1[i][j]=random(8);

weight_norm();
normalization();
}


void forword_respond()
{
int   i,j,k;
int   maxneuron;
float S;
float Smax;
```

```
for(k=0; k<image; k++)
  {
  printf("\nImage %d\n", k+1);
  Smax=0;
  maxneuron=0;
  for(i=0; i<outneuron; i++)
    {
    S=0;
    for(j=0; j<inneuron; j++) S=S+W1[i][j]*X1[k][j];
    if(S>Smax) { Smax=S; maxneuron=i; }
    }
  for(i=0; i<outneuron; i++)
    {
    X2[i]=0;
    if(i==maxneuron) X2[i]=1;
    printf(" %d", X2[i]);
    }
  printf("   %f", Smax);
  }
}

void separation()
{
int   i,k,l;
float d,dd;


for(k=0; k<outneuron; k++)
  for(l=0; l<outneuron; l++)
  {  d=0;
     for(i=0; i<inneuron; i++)
      { dd=W1[k][i]-W1[l][i];
        d=d+dd*dd;
      }
     DWW[k][l]=d;
  }

for(k=0; k<outneuron; k++)
  for(l=0; l<image; l++)
  {  d=0;
     for(i=0; i<inneuron; i++)
      { dd=W1[k][i]-X1[l][i];
        d=d+dd*dd;
      }
     DWX[k][l]=d;
  }

}


void learning()
{
int   i,j,l;
int   pn;
float f;

float F[outneuron][inneuron];

separation();
pn=random(4);

for(i=0; i<outneuron; i++)
```

```
    for(l=0; l<inneuron; l++)
    {  F[i][l]=0;
       for(j=0; j<outneuron; j++)
       { if (j!=i) { f=(W1[i][l]-W1[j][l])/DWW[i][j];
                F[i][l]=F[i][l]+f;
                }
       }

       f=(X1[pn][l]-W1[i][l])/DWX[i][pn];
       F[i][l]=F[i][l]+mass*f;

    }

for(i=0; i<outneuron; i++)
  for(l=0; l<inneuron; l++)  W1[i][l]=(1-T)*W1[i][l]+T*F[i][l];

}


void print_weight()
{
int   i,j,k;

for(i=0; i<outneuron; i++)
  {
   for(j=0; j<inneuron; j++)
   printf(" %f", W1[i][j]);
   printf("\n");
  }
}


void weight_norm()
{
int   i,j;

for(i=0; i<outneuron; i++)
  {
  WN[i]=0;
  for(j=0; j<inneuron; j++)  WN[i]=WN[i]+W1[i][j]*W1[i][j];
  WN[i]=sqrt(WN[i]);
  }
}


void normalization()
{
int   i,j;
float norm;

for(i=0; i<outneuron; i++)
  {
  norm=WN[i];
  for(j=0; j<inneuron; j++) W1[i][j]=W1[i][j]/norm;
  }
}
```

```
void normalize_image()
{
int   i,j;
float       norm;

for(i=0; i<image; i++)
  {
  IL[i]=0;
  for(j=0; j<inneuron; j++) IL[i]=IL[i]+X1[i][j]*X1[i][j];
  IL[i]=sqrt(IL[i]);
  }

for(i=0; i<image; i++)
  {
  norm=IL[i];
  for(j=0; j<inneuron; j++) X1[i][j]=X1[i][j]/norm+0.05;

  }
}

void chaos()
{
int   n1,n2;

n1=random(inneuron);
n2=random(outneuron);
W1[n2][n1]=0;
}
```

< APPENDIX A(ii) >

```
/*-------------------------------------------------------------------------
      MCL.C 6-JAN-92

      Modified Compeititve Learning

      - The rule of response in the second layer is followed
        the competitive learning.
      - The weight updating rule is a new, not the same as
        the classic one. Its idea is based on a phenomena in
        electrostatics, like pole repell and unlike pole attract.
-------------------------------------------------------------------------*/

#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<conio.h>
#include<graphics.h>

#define iteration 4096
#define inneuron 48
#define outneuron 8
#define image     6
#define     T     0.0001
#define mass      25

int   X2[outneuron];
float X1[image][inneuron];
float DWW[outneuron][outneuron];
float DWX[outneuron][image];
float IL[image];
float W1[outneuron][inneuron];
float WN[outneuron];

void  forword_respond(void);
void  inputpattern(void);
void  learning(void);
void  loadinput(void);
void  normalization(void);
void  normalize_image(void);
void  testing(void);
void  weight_init(void);
void  weight_norm(void);


main()
{
int   iterate;
int   index;
void  print_weight();

loadinput();
normalize_image();
weight_init();
for(iterate=0; iterate<iteration; iterate++)
  {
/*  printf(".", iterate);*/
  learning();
  weight_norm();
  normalization();
  index=iterate%256;
```

```
  if (index==0) { printf("\n\n --- %d ---\n\n",iterate);
                  print_weight();
                  forword_respond();
                  printf("\n\n");   }
  }
printf("\n\n --- %d ---\n\n",iterate);
print_weight();
forword_respond();
}

void loadinput()
{
char  xc;
int   i,j,k;
int   xx;
FILE  *input;

/* input=fopen("mc1.dat","r"); */
input=fopen("char01.dat","r");
for(k=0; k<image; k++)
  {  for(i=0; i<inneuron; i++)
     {   xc=getc(input);
       xx=atoi(&xc);
       X1[k][i]=xx;
     }
     getc(input);
  }
fclose(input);
}

void weight_init()
{
int   i,j;

for(i=0; i<outneuron; i++)
  for(j=0; j<inneuron; j++) W1[i][j]=random(8);

weight_norm();
normalization();
}


void forword_respond()
{
int   i,j,k;
int   maxneuron;
float S;
float Smax;

for(k=0; k<image; k++)
  {
  printf("\nImage %d\n", k+1);
  Smax=0;
  maxneuron=0;
  for(i=0; i<outneuron; i++)
    {
    S=0;
    for(j=0; j<inneuron; j++) S=S+W1[i][j]*X1[k][j];
    if(S>Smax) { Smax=S; maxneuron=i; }
    }
  for(i=0; i<outneuron; i++)
```

```
    {
    X2[i]=0;
    if(i==maxneuron) X2[i]=1;
    printf(" %d", X2[i]);
    }
  printf("   %f", Smax);
  }
}

void separation()
{
int   i,k,l;
float d,dd;


for(k=0; k<outneuron; k++)
  for(l=0; l<outneuron; l++)
  {  d=0;
     for(i=0; i<inneuron; i++)
      { dd=W1[k][i]-W1[l][i];
        d=d+dd*dd;
      }
     DWW[k][l]=d;
  }

for(k=0; k<outneuron; k++)
  for(l=0; l<image; l++)
  {  d=0;
     for(i=0; i<inneuron; i++)
      { dd=W1[k][i]-X1[l][i];
        d=d+dd*dd;
      }
     DWX[k][l]=d;
  }

}

void learning()
{
int   i,j,l;
int   pn;
float f;

float F[outneuron][inneuron];

separation();
pn=random(4);

for(i=0; i<outneuron; i++)
    for(l=0; l<inneuron; l++)
    {  F[i][l]=0;
       for(j=0; j<outneuron; j++)
       { if (j!=i) { f=(W1[i][l]-W1[j][l])/DWW[i][j];
                     F[i][l]=F[i][l]+f;
                     }
       }

       f=(X1[pn][l]-W1[i][l])/DWX[i][pn];
       F[i][l]=F[i][l]+mass*f;

    }
```

```
for(i=0; i<outneuron; i++)
  for(l=0; l<inneuron; l++)  W1[i][l]=(1-T)*W1[i][l]+T*F[i][l];

}

void print_weight()
{
int   i,j,k;

for(i=0; i<outneuron; i++)
  {
   for(j=0; j<inneuron; j++)
   printf(" %f", W1[i][j]);
   printf("\n");
  }
}

void weight_norm()
{
int   i,j;

for(i=0; i<outneuron; i++)
  {
  WN[i]=0;
  for(j=0; j<inneuron; j++)  WN[i]=WN[i]+W1[i][j]*W1[i][j];
  WN[i]=sqrt(WN[i]);
  }
}


void normalization()
{
int   i,j;
float norm;

for(i=0; i<outneuron; i++)
  {
  norm=WN[i];
  for(j=0; j<inneuron; j++) W1[i][j]=W1[i][j]/norm;
  }
}

void normalize_image()
{
int   i,j;
float       norm;

for(i=0; i<image; i++)
  {
  IL[i]=0;
  for(j=0; j<inneuron; j++) IL[i]=IL[i]+X1[i][j]*X1[i][j];
  IL[i]=sqrt(IL[i]);
  }

for(i=0; i<image; i++)
  {
  norm=IL[i];
  for(j=0; j<inneuron; j++) X1[i][j]=X1[i][j]/norm+0.05;
  } }
```

< APPENDIX A(iii) >

```
/*-------------------------------------------------------------------------
     MPR01.C          28-JAN-92

     Machine Part Recognition (Part 1)

     Function : Parts Classification
                Reject Defective Parts
                Testing

     Input    : MPRG01.DAT
                MPRD01.DAT
-------------------------------------------------------------------------*/

#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<conio.h>
#include<graphics.h>

#define iteration 128
#define inneuron 48
#define outneuron 2
#define image    2
#define defnum   2
#define T        0.0001
#define t        0.01
#define p        0.05
#define level1   1
#define mass     25

int     Super;
int     X2[outneuron];
int     X3[outneuron];
int     X4[outneuron];
int     X5[outneuron];
float   X1[image][inneuron];
float   TI[inneuron];
float   DWW[outneuron][outneuron];
float   DWX[outneuron][image];
float   IL[image];
float   W1[outneuron][inneuron];
float   W2[outneuron][inneuron];
float   WN[outneuron];

void    autotesting(void);
void    forword_respond(void);
void    GenerateX2(void);
void    GenerateX3(void);
void    GenerateX4(void);
void    GenerateX5(void);
void    inputpattern(void);
void    learn12(void);
void    learn13(void);
void    loaddefect(void);
void    loadinput(void);
void    normalization(void);
void    normalize_image(void);
void  normalize_W2(void);
void    testing(void);
void    testinput(void);
```

```
void    weight_init(void);
void    weight_norm(void);

main()
{
int     i,j;
int     defno;
int     iterate;
int     index;
void    print_weight();

Super=0;
loadinput();
printf(" loadinput completed ...\n");
normalize_image();
printf(" normalization completed ...\n");
weight_init();
printf(" unsupervised learning started ...\n");
for(iterate=0; iterate<iteration; iterate++)
  {
  learn12();
  weight_norm();
  normalization();
  index=iterate%128;
  if (index==0) { printf("\n --- %d ---\n",iterate);
              forword_respond(); }
  }
printf("\n --- %d ---",iterate);
forword_respond();
printf("\n\n unsupervised learning completed ...\n");
printf(" loading defective pattern ...\n");
loaddefect();
printf(" normalize defective pattern ...\n");
normalize_image();
printf(" supervised learning started - learn defectvie item ...\n");
/*  printf(" defective item = %d\n", defnum);  */
Super=1;
for(iterate=0; iterate<iteration; iterate++)
  {
  defno=random(defnum);
  printf("\n%d ...\n", defno);
  for(i=0; i<inneuron; i++) TI[i]=X1[defno][i];
  GenerateX2();
  GenerateX4();
  learn13();
  GenerateX3();
  GenerateX5();
    printf("\n");
    for(j=0; j<outneuron; j++) printf("%d", X2[j]);
    printf("   ");
    for(j=0; j<outneuron; j++) printf("%d", X3[j]);
    printf("   ");
    for(j=0; j<outneuron; j++) printf("%d", X4[j]);
    printf("   ");
    for(j=0; j<outneuron; j++) printf("%d", X5[j]);
  }
printf("\n supervised learning completed ... \n\n");
printf(" Auto-Testing Started \n");
normalize_W2();
autotesting();
getchar();
}
```

```
void loadinput()
{
char    xc;
int     i,j,k;
int     xx;
FILE    *input;

input=fopen("mprg01.dat","r");
for(k=0; k<image; k++)
  {  for(i=0; i<inneuron; i++)
     {   xc=getc(input);
       xx=atoi(&xc);
       X1[k][i]=xx;
     }
     getc(input);
  }
fclose(input);
}


void loaddefect()
{
char    xc;
int     i,j,k;
int     xx;
FILE    *defect;

defect=fopen("mprd01.dat","r");
for(k=0; k<defnum; k++)
  {  for(i=0; i<inneuron; i++)
     {  xc=getc(defect);
      xx=atoi(&xc);
      X1[k][i]=xx;
     }
     getc(defect);
  }
fclose(defect);
}


void weight_init()
{
int     i,j;

for(i=0; i<outneuron; i++)
  for(j=0; j<inneuron; j++) W1[i][j]=random(8);
weight_norm();
normalization();

for(i=0; i<outneuron; i++)
  for(j=0; j<inneuron; j++) W2[i][j]=0;
}


void forword_respond()
{
int     i,j,k;
int     maxneuron;
float   S;
float   Smax;
```

```
for(k=0; k<image; k++)
  {
  printf("\nImage %d\n", k+1);
  Smax=0;
  maxneuron=0;
  for(i=0; i<outneuron; i++)
    {
    S=0;
    for(j=0; j<inneuron; j++) S=S+W1[i][j]*X1[k][j];
    printf(" %f", S);
    if(S>Smax) { Smax=S; maxneuron=i; }
    }
  for(i=0; i<outneuron; i++)
    {
    X2[i]=0;
    if(i==maxneuron) X2[i]=1;
    }
  }
}


void separation()
{
int     i,k,l;
float   d,dd;


for(k=0; k<outneuron; k++)
  for(l=0; l<outneuron; l++)
  {  d=0;
     for(i=0; i<inneuron; i++)
      { dd=W1[k][i]-W1[l][i];
        d=d+dd*dd;
      }
     DWW[k][l]=d;
  }

for(k=0; k<outneuron; k++)
  for(l=0; l<image; l++)
  {  d=0;
     for(i=0; i<inneuron; i++)
      { dd=W1[k][i]-X1[l][i];
        d=d+dd*dd;
      }
     DWX[k][l]=d;
  }

}


void learn12()
{
int     i,j,l;
int     pn;
float   f;

float   F[outneuron][inneuron];

separation();
pn=random(image);

for(i=0; i<outneuron; i++)
```

```
    for(l=0; l<inneuron; l++)
    {  F[i][l]=0;
       for(j=0; j<outneuron; j++)
       { if (j!=i) { f=(W1[i][l]-W1[j][l])/DWW[i][j];
                F[i][l]=F[i][l]+f;
                }
       }

       f=(X1[pn][l]-W1[i][l])/DWX[i][pn];
       F[i][l]=F[i][l]+mass*f;

    }


for(i=0; i<outneuron; i++)
  for(l=0; l<inneuron; l++)  W1[i][l]=(1-T)*W1[i][l]+T*F[i][l];

}


void learn13()
{
int     i,j;

for(i=0; i<outneuron; i++)
  for(j=0; j<inneuron; j++) W2[i][j]=(1-t)*W2[i][j]+p*TI[j]*X4[i];
}


void weight_norm()
{
int     i,j;

for(i=0; i<outneuron; i++)
  {
  WN[i]=0;
  for(j=0; j<inneuron; j++)  WN[i]=WN[i]+W1[i][j]*W1[i][j];
  WN[i]=sqrt(WN[i]);
  }
}


void normalize_W2()
{
int   i,j;

for(i=0; i<outneuron; i++)
  { WN[i]=0;
    for(j=0; j<inneuron; j++) WN[i]=WN[i]+W2[i][j]*W2[i][j];
    WN[i]=sqrt(WN[i]);
  }

for(i=0; i<outneuron; i++)
  for(j=0; j<inneuron; j++)
     { if(WN[i]!=0) W2[i][j]=W2[i][j]/WN[i]; }
}
```

```
void normalization()
{
int     i,j;
float   norm;

for(i=0; i<outneuron; i++)
  {
  norm=WN[i];
  for(j=0; j<inneuron; j++) W1[i][j]=W1[i][j]/norm;
  }
}

void normalize_image()
{
int     i,j;
float   norm;

for(i=0; i<image; i++)
  {
  IL[i]=0;
  for(j=0; j<inneuron; j++) IL[i]=IL[i]+X1[i][j]*X1[i][j];
  IL[i]=sqrt(IL[i]);
  }

for(i=0; i<image; i++)
  {
  norm=IL[i];
  for(j=0; j<inneuron; j++) X1[i][j]=X1[i][j]/norm+0.005;

  }
}


void testinput()
{
int     j;
float   TIL;

TIL=0;
for(j=0; j<inneuron; j++) TIL=TIL+TI[j]*TI[j];
TIL=sqrt(TIL);
for(j=0; j<inneuron; j++) TI[j]=TI[j]/TIL+0.05;
}


void GenerateX2()
{
int     i,j;
int     Winner;
float   Smax,S;

Smax=0;
for(i=0; i<outneuron; i++)
  { S=0;
    for(j=0; j<inneuron; j++) S=S+W1[i][j]*TI[j];
    if(S>Smax) { Smax=S; Winner=i; }
  }
for(i=0; i<outneuron; i++)
  { X2[i]=0;
    if(i==Winner) X2[i]=1;
  }
```

```
}

void GenerateX3()
{
int      i,j;
float    S;

for(i=0; i<outneuron; i++)
  { S=0;
    for(j=0; j<inneuron; j++)  S=S+W2[i][j]*TI[j];
    X3[i]=0;
    if(S>=level1) X3[i]=1;
  }
}


void GenerateX4()
{
int      i,j;
float    S;

for(i=0; i<outneuron; i++)
  { X4[i]=0;
    if((Super==1)&&(X2[i]==1)) X4[i]=1;
  }
}


void GenerateX5()
{
int      i,j;

for(i=0; i<outneuron; i++)  X5[i]=X2[i]-X3[i];
}


void autotesting()
{
int      i,j,k;

loadinput();
normalize_image();
Super=0;
for(k=0; k<image; k++)
  {  for(i=0; i<inneuron; i++) TI[i]=X1[k][i];
     GenerateX2();
     GenerateX4();
     GenerateX3();
     GenerateX5();
      printf("\n");
      for(j=0; j<outneuron; j++) printf("%d", X2[j]);
      printf("   ");
      for(j=0; j<outneuron; j++) printf("%d", X3[j]);
      printf("   ");
      for(j=0; j<outneuron; j++) printf("%d", X4[j]);
      printf("   ");
      for(j=0; j<outneuron; j++) printf("%d", X5[j]);
  }

loaddefect();
normalize_image();
```

```
Super=1;
for(k=0; k<image; k++)
  {  for(i=0; i<inneuron; i++) TI[i]=X1[k][i];
     GenerateX2();
     GenerateX4();
     GenerateX3();
     GenerateX5();
      printf("\n");
      for(j=0; j<outneuron; j++) printf("%d", X2[j]);
      printf("   ");
      for(j=0; j<outneuron; j++) printf("%d", X3[j]);
      printf("   ");
      for(j=0; j<outneuron; j++) printf("%d", X4[j]);
      printf("   ");
      for(j=0; j<outneuron; j++) printf("%d", X5[j]);
  }
}
```

< APPENDIX A(iv) >

Here is the content of the file CHAR01.DAT (for use in MCL.C / CMCL.C)

```
11110000110001001001001011111111001110000110000l
11111011001111001111111011111011001111001111111O
01111111111111000011000011000011000011111111011111
11111111111111000011111111111111000011111111111111
00111001001001001001001001001001001011001011000l111
11001111001111001111111111111111111001111001111001l
```

Here is the content of the file CHAR02.DAT. The last two lines are dummy. (for use in MCL.C / CMCL.C)

```
11111010000110000110001011110010011010001110001l
11110010011010001010001011111010000010000010000O
11110000110001001001001011111111001110000110000l
11111011001111001111111011111011001111001111111O
01111111111111000011000011000011000011111111011111
11111111111111000011111111111111000011111111111111
00111001001001001001001001001001001011001011000l111
11001111001111001111111111111111111001111001111001l
```

Here is the content of MPRG01.DAT (good machine part for MPR.C)

```
11111111001111001110000110000111001111001111111l
00000000110000110001111001111000110000110000000O
```

Here is the content of MPRD01.DAT (defective machine part for MPR.C)

```
11011111001111001110000110000011001111001111111l
00010000110000110001111001111000110000110000l000
```

< *APPENDIX B: Bibilography* >

Amari S-I (1977). Neural theory of association and concept-formation. Biological Cybernetics 26, pp175-185, 1977.

Amari S-I (1988). Associative Memory and Its Statistical Neurodynamical Analysis. In Haken (1988).

Anderson J.A. (1983). Cognitive and Psychological Computation with Neural Models. IEEE Transactions on System, Man, and Cybenetics, SMC-13, 799-815, 1983.

Anderson J.A. and Rosenfeld E. (1988). Neurocomputing, Foundation of Research. MIT Press, 1988.

Bachmann C.M., Cooper L.N., Dembo A. and Zeitouni O. (1987). A Relaxation Model for Memory with High Storage Density. Proceedings of the National Academy of Sciences, USA 84, 7592-7531, 1987.

Carpenter G.A. and Grossberg S. (1986). Neural Dynamics of Category Learning and Recognition: Attention, Memory Consolidation, and Amnesia, in Brain Structure, Learning, and Memory. edited by J. Davis, R. Newburgh and E. Wegman, AAAs Symposium series, 1986.

Carpenter G.A. and Grossberg S. (1987a). A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine. Computer Vision, Graphics and Image Processing 37, 1987.

Carpenter G.A. and Grossberg S. (1987b). ART 2: self-organization of stable category recognition codes for analog input pattern. Applied Optics, VOL. 26, NO.23, 4919-4946, 1987.

Carpenter G.A. et. al. (1987). Technical Comments on Computer with Neural Networks. Science, VOL. 235, 1226-1229, 1987.

Carpenter G.A. and Grossberg S. (1988). The ART of Adaptive Pattern Recognition by a Self-Organization Neural Network. IEEE Computer, Vol 21, No 3, March, 77-88, 1988.

Churchland P.S., Koch C. and Sejnowski T.J. (1990). What Is Computational Neuroscience? In Computational Neuroscience, edited by Eric L. Schwartz, Chapter 5. 1990.

Cohen M.A. and Grossberg S. (1983). Absolute Stability of Global Pattern Formation and Parallel Memory Storage by Competitive

Neural Networks. IEEE Transactions on System, Man and Cybernetics, SMC-13, 815-826, 1983.

Griffith J.S. (1971). Mathematical Neurobiology. An Introduction to the Mathematics of the Nervous System. Academic Press, 1971.

Grossberg S. (1976). Adaptive Pattern Classification and Universal Recording: I. Parallel Development and Coding of Neural Feature Detectors. Biological Cybernetics 23: 121-134, 1976.

Grossberg S. (1980a). How does a brain build a cognitive code. Psychological Review 87:1-51, 1980.

Grossberg S. (1980b). Mathematical Psychology and Psychophysiology. American Mathematical Society, 1980.

Grossberg S. (1980c). Adaptive resonance in development, perception and cognition. In Mathematical Psychology and psychophysiology, edited by Grossberg S., AMS, 1980.

Grossberg S. (1987). Competitive Learning: From Interactive Activation to Adaptive Resonance. Cognitive Science 11, 23-63, 1987.

Hebb D.O. (1949). The Organization of Behavior, Wiley, 1949. Introduction and Chapter 4 are also appeared in Neurocomputing, edited by J.A.Anderson and E.Rosenfeld, 45-56, MIT Press, 1988.

Hebb D.O. (1972). Textbook of Psychology. 3rd ed. Saunders Company, 1972.

Hebb D.O. (1980). Essay on Mind. Lawerence Erlbaum Associate. 1980.

Hecht-Nielsen R. (1990). Neurocomputing. Addison Wesley, 1990.

Hertz J., Krogh A. and Palmer R.G. (1991). Introduction to the Theory of Neural Computation. Addison-Wesley. 1991.

Hopfield J.J. (1982). Neural networks and physical systems with emergent collective computational abilities. Proceesings of the National Academy of Sciences 79:2554-2558. 1982.

Hopfield J.J. (1984). Neurons with graded response have collective computational properties like those of two-state neurons. Proceedings of the National Academy of Sciences USA 81, May 1984, 3088-3092, 1984.

Hui C.C. (1991). An Error Correcting Algorithm for Hopfield Network. (Unpublished paper from CUHK) 1991.

IEEE (1992). Special Issues on Neural Networks in Control Systems. IEEE Control Systems, Vol. 12, No. 2, April, 1992.

Kandel R.E. and Schwartz H.J. (1982). Molecular Biology of Learning: Modulation of Transmitter Release. Science, Vol 218,

433-443, 1982.

Kandel R.E. and Schwartz H.J. (1985). Principles of Neural Sciences, 2nd ed., Elsevier, 1985.

Kosko B. (1992). Neural Networks and Fuzzy Sysytems. Prentice Hall, 1992.

Lee C.K., Sum P.F. and Tam P.K. (1992a). An Unsupervised Learning Algorithm for Character Recognition. (To be presented in IJCNN92, Baltimore, USA, June 8-11, 1992).

Lee C.K., Sum P.F. and Tam P.K. (1992b). Hierarchical Neural Network for Machine Part Recognition. (To be presented in IEEE ETFA'92, Melbourne, Australia, August 11-14, 1992)

Lippamnn R.P. (1988). An Introduction to computing with Neural Nets. in Neural Networks, Artificial Neural Networks: Theoretical Concepts, edited by V. Vemuri. IEEE Computer Society Press Technology Series, 1988.

McCleland J.L. and Rumelhart D.E. (1981). An Interactive Model of Context Effects in Letter Perception: Part 1. An Account of Basic Finding. Psychological Review, 88, 375-407, 1981.

McCleland J.L. and Rumelhart D.E. (1985). Distributed Memory and the Representation of Genral and Specific Information. Journal of Experimental Psychology: General. 159-188, 1985.

McCleland J.L. and Rumelhart D.E. (1987). Parallel Distribution Processing. Explorations in the Microstructure of Cognition. Volume 2: Psychological and Biological and Models.

McCulloch W.S. and Walter P. (1943). A Logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics 5:115-133. 1943.

Nguyen D.H. and Widrow B. (1990). Neural Networks for Self-Learning Control System. IEEE Control Systems Magazine (April), 18-23, 1990.

Pinel J.P.J. (1990). Biopyschology. Allyn & Bacon. 1990.

Pitts W. and McCulloch W.S. (1947). How we know universals: the perception of auditory and visaul forms. Bulletin of Mathematical Biophysics 9:127-147, 1947.

Rosenblatt F. (1958). The Perceptron: The Probabilistic Model for Information Storage and Organization in the Brain. Psychological Review, 65, 386-408, 1958.

Rosenblatt F. (1960). Perceptual Generalization over Transformation Groups. in Proceeding of an Interdisciplinary Conference: Self-Organizing Systems. edited by M.C. Yovits and C.Cameron, Pergamon Press, 1960.

Rosenblatt F. (1962). Principles of Neurodynamics : Perceptrons and the Theory of Brain Mechanisms. Spartans Books. 1962.

Rosenblatt F. (1964). A Model for Experiential Storage in Neural Network. in Computer and Information Science, edited by J.T. Tou and R.H. Wilcox. Spartan, 1964.

Rosenblatt F., Farrow J.T. and Herblin W.F. (1966). Tranfer of Conditioned Responses from Trained Rats to Untrained Rats by means of a Brain Extract. Nature, Jan, 46-48, 1966.

Rosenblatt F. (1967). Recent Work on Theoretical Models of Biological Memory. in Computer and Information Science-II, edited by J.T. Tou, Academic Press, 1967.

Rumelhart D.E. and McCleland J.L. (1982). An Interactive Activation Model of Context Effects in Letter Perception: Part 2. The contextual Enhancement Effect and Some Tests and Entensions of the Model. Psychological Review, Vol 89, 6094, 1982.

Rumelhart D.E. and Zipser D. (1985). Feature Discovery by Competitive Learning. Cognitive Science 9, 75-112, 1985.

Rumelhart D.E. and McClelland J.L. (1987). Parallel Distributed Processing. Explorations in the Microstructure of Cognition. Volume 1: Foundation.

Tou J.T. and Wilcox R.H. (1964). Computer and Inforamtion Science. Spartan, 1964.

Tou J.T. (1967). Computer and Information Science-II, Academic Press, 1967.

Sum P.F., Lee C.K. and K.S.Tam (1992), Machine Part Recognition Using Layer Nets. (To be presented in Automation'92, Taiwan, July 1-4, 1992).

Wasserman P.D. (1989). Neural Computing. Theory and Practice. van Nostrand Reinhold, 1989.

Yovits M.C. and Cameron C. (1960). Self-Organization Systems. Pergamon Press, 1960.

Further References

Abu-Mostafa Y.S. and Jacques J.-M. ST. (1985). Information Capaticity of the Hopfield Model. IEEE Transaction on Information Theory, Vol. IT-31, NO.4, 461-464. 1985.

Ackley D.H., Hinton G.E. and Sejnowski T.J. (1985). A Learning Algorithm for Boltzmann Machine. Cognitive Science 9, 147-169, 1985.

Ahmed N. and Rao K.R. (1975). Orthogonal Transforms for Digital Signal Processing, Springer-Verlag,1975.

Aiyer S.V.B., Niranjan M. And Fallside F. (1990). A Theoretical Investigation into the Perfromance of the Hopfield Model. IEEE Transaction on Neural Networks, Vol 1, No 2, 204-215, 1990.

Albus J.S. (1991). Outline for a Theory of Intelligence. IEEE Transaction on SMC, vol 21, No 3, 473-509. 1991.

Andrews H.C. (1972). Introduction to Mathematical Techniques in Pattern Recognition, John Wiley and Sons, 1972.

Arbib M.A. (1987). Brains, Machines and Mathematics., 2nd ed. Springer-Verlag, 1987.

Atkinson R.L., Atkinson R.C., Smith E.E., Bem D.J. and Hilgard E.R. (1990). Introduction to Psychology. 10th ed. Harcourt Brace Jovanovich Publishers. 1990

Bacon W.F. and Egeth H.E. (1991). Local Processes in Preattentive Feature Detection. Journal of Experimental Psychology. Learning, Memory and Cognition, 77-90, 1991.

Barto A.G. and Anandan P. (1985). Pattern-Recognizing Stochastic Learning Automata. IEEE Transactions on System, Man and Cybernetics, SMC-15, 360-375, 1985.

Block H.D. (1962). The Perceptron: A model for Brain Functioning I. Review of Modern Physics, Vol 34, 123-135, 1962.

Block H.D., Knight B.W.Jr and Rosenblatt F. (1962). Analysis of a Four-Layer Series-Coupled Perceptron II. Review of Modern Physics. Vol 34, 135-142, 1962.

Block H.D., Nilsson N.J. and Duda R.O. (1964). Determination and Direction of Features in Patterns. in Computer and Information Science, edited by J.T. Tou and R.H. Wilcox, Spartan, 1964.

Burke L.I. (1991). Clustering Charaterization of Adaptive Resonance. Neural Networks, Vol 4, pp. 485-491, 1991.

Bullock D. and Grossberg S. (1988). Self-Organizing Neural Architectures for Eye Movements, Arm Movements and Eye-Arm Coordination. In Haken (1988). (The term circular reaction has been quoted. Also a book called "The origins of intelligence in children", which is edited by Prof. Jean Piaget, has also been included in References.)

Carpenter G.A. and Grossberg S. (1988). Self-Organizing Neural Network Architectures for Real Time Adaptive Pattern Recognition. In Haken (1988).

Carpenter G.A., Grossberg S. and Rosen D.B. (1991). ART 2-A: An Adaptive Resonace Alogrithm for Rapid Category Learning and Recognition. Neural Networks, Vol 4, pp. 493-504. 1991.

Caudill Maureen (1991). Expert Networks. Byte October, 108-116, 1991.

Cesari L. (1971). Asymptotic Behavior and Stability Problems in Ordinary Differential Equation. Springer-Verlag, 1971.

Chan L.W. (1990). Efficacy of Different Learning Algorithms of the Back Propagation Network. Proceeding of IEEE Region 10 Conference on Computer and Communication System, 23-27, 1990.

Changeux J.P and Dehaene S. (1989). Neuronal Model of cognitive Functions. Cognition 33, 63-109, 1989.

Cheung C.W. (1990). The Study of the Application of Neural Networks. Unpublished Final Year Project Report in Dept. of Electronics Enginnering, code 7a90, 1990.

Coburn H.E. (1951). The Brain Analogy. Psyschological Review, 155-178, 1951.

Coburn H.E. (1953a). The Brain Analogy: Transfer of Differentiation. Psychological Review, 413-422, 1953.

Coburn H.E. (1953b). The Brain Analogy: Association Tracts. Psychological Review, 197-208, 1953.

Coddington E.A. (1974). An Introduction to Ordinary Differential Equation. Prentice Hall, 1974.

Davis G.W.Jr. Sensitivity Analysis in Neural Net Solutions. IEEE Transaction on SMS, Vol 19, No 5, 1978-1082, 1989.

Domany E., van Hemmen J.L. and Schulten K. (1991). Model of Neural Networks. Springer-Verlag, 1991.

Edelman G.M. (1989). Neural Darwinism. The Theory of Neuronal Group Selection. Oxford University Press, 1989.

Farah M.J. and McClelland J.L. (1991). A Computational Model of Semantic Memory Impairment : Modality Specificity and Emergent Category Specificity. Journal of Experimental Psychology: General, Vol 120, No 4, 339-357, 1991.

Geman S. and Geman D. (1984). Stochastic Relaxation, Gibbs Distribution and the Bayesian Restoration of Image. IEEE Transactions on System, Man and Cybernetics, SMC-14, 721-741, 1984.

Gonzalez R.C. and Wintz P. (1987). Digital Image Processing, Addison-Wesley, 1987.

Grimsdale R.L. , Sumner F.H., Tunis C.J. and Kilburn T. (1959). A System for the Automatic Recognition of Patterns. Proceeding of IEE, 106 Part B, 210-221, 1959.

Grossberg S. and Kuperstein M. (1989). Neural Dynamics of

Adaptive Sensory-Motor Control, Pergamon, 1989.

Guyon I., Alberecht P., Le Cun Y., Denker J. and Hubbard W. (1991). Design a Neural Network Charater Recognizer for a Touch Terminal. Pattern Recognition, Vol 24, No. 2, 105-119, 1991.

Haken H. (1988). Neural and Synergetic Computer. Springer-Verlag, 1988.

Hale J.K. and LaSalle J.P. (1967). Differential Equation and Dynamic Systems. Academic Press, 1967.

Haykin S. (1984). Introduction to Adaptive Filter. Macmillan Publishing Company. 1984.

Hendler J.A. (1989) Marker-Passing over Microfeatures: Towards a Hybrid Symbolic/Connectionist Model. Cognitive Science 13, 79-106, 1989.

Hinton G.E. and Sejnowski T.J. (1983). Optimal Perceptual inference. Proceeding of IEEE Conference on Pattern Recognition and computer Vision, 448-453, 1983.

Hopfield J.J. and Tank D.W. Computing with Neural Circuits: A Model. Science 233, 625-632.

Hulse S.H., Egeth H. and Deese J. (1980). The Psychology of Learning. 5th ed. McGraw Hill. 1980.

IEEE (1990a). Special Issue on Neural Networks I. Proceedings of IEEE, VOL. 78, NO.9, 1990.

IEEE (1990b). Special Issue on Neural Networks II. Proceedings of IEEE, VOL. 78, NO.9, 1990.

IEEE (1991). Special Issue on Aritificial Neural Networks. IEEE Transaction on Computers Vol 40, No 12, 1991.

Ikeda N. and Torioka T. A Model of Associative Memory Based on Adaptive Feature-Detecting Cells. IEEE Transactions on System, Man and Cybernetics, Vol. 20, No. 2, 436-443. 1990.

Jacobs R.A., Jordan M.I. and Barto A. (1991). Task Decomposition Through Competition in a Modular Connectionist Architecture: The What and Where Vision Tasks. Cognitive Science 15, 219-250. (1991).

Jordan D.W. and Smith P. (1977). Nonlinear Ordinary Differential Equations. Oxford University Press, 1977

Kan W.K., Wong K.H. and Law H.M. (1990). Non-Overlapped Trees of Probabilistic Logic Neuron. Proceeding of the IEEE Region 10 Conference on Computer and Communication Systems, 37-39, 1990.

Kohonen T. (1977). Associative Memory. A system-theoritical approach. Springer-Verlag. 1977.

Kohonen T. (1982). Self-organized formation of topologically correct feature maps. Biological Cybernetics 43:59-69, 1982.

Kohonen T. (1988). Self-Organization and Associative Memory. 2nd ed. Springer-Verlag. 1988.

Kong S-G. and Kosko B. (1991). Differential Competitive Learning for Centroid Estimation and Phoneme. IEEE Transaction on Neural Networks, VOL. 2, 118-124. 1991.

Kosko B. (1987). Adaptive bidirectional associative memories. Applied Optics, VOL. 26, NO. 23, 4947-4960, 1987.

Kosko B. (1990). Unsupervised Learning in Noise. IEEE Transaction on Neural Networks, VOL. 1, NO. 1, 44-57, 1990.

Kosko B. (1991). Stochastic Competitive Learning. IEEE Transaction on Neural Networks, VOL. 2, 522-529, 1991.

LaSalle J.P. (1967). An Invariance Principle in the Theory of Stablity. In Differential Equation and Dynamic Systems, edited by J.K.Hale and J.P.LaSalle. Academic Press, 1967.

McEliece R.J., Posner E.C., Rodemich E.R. and Venkatesh S.S. (1987). The Capacity of the Hopfield Associative Memory. IEEE Transaction on Information Theory, VOL. IT-33, NO.4, 461-482, 1987.

Minsky M.L. (1967). Computation: Finite and Infinite Machine. Prentice Hall. 1967.

Minsky M, Seymour P. (1972). Perceptrons : An Essay in Computational Geometry. MIT Press. (1972)

Milner P.M. (1957). The Cell Assembly: MARK II. Psychology Review, 64, 242-252, 1957.

Pao Y.H. (1989). Adaptive Pattern Recognition and Neural Network, Addison-Wesley, 1989.

Scalero R.S. and Tepedelenlioglu N. (1992). A Fast New Algorithm for Training Feedforward Neural Networks. IEEE Transactions on Signal Processing, vol 40, No 1, 202-210, 1992.

Shapiro S.S. (1987). Encyclopedia of Aritificial Intelligence. Vol 2. Wiley. 1987.

Shavlik J.W., Mooney R.J. and Tonell G.G. (1991). Symbolic and Neural Learning Algorithms: An Experimental Comparison. Machine Learning, 111-143, 1991.

Tank D.W. and Hopfield J.J. (1986). Simple Neural Optimization Networks: An A/D converter, Signal Decision Circuit and a Linear Programming Circuit. IEEE Transactions on Circuits and System. 533-541, 1986.

Taylor W.K. (1959). Pattern Recognition by Means of Automatic Analogue Apparatus. Proceeding of IEE, 106 Part B, 198-209, 1959.

Time-Life Books (1989). Alternative Computer, 1989.

Torioka T. and Ikeda N. (1990). Consideration on Pattern-Separating Function in a Generalized Random Nerve Net Consisting of Two Layers. IEEE Transaction on SMC, VOL. 20, NO. 3, 619-627, 1990.

von der Malsburg Chr. (1973). Self-organization of orientation sensitive cells in the striata cortex. Kybernatik 14:85-100. 1973.

Vemuri V. (1988). Aritificial Neural Networks: An Introsuction. in Neural Networks, Artificial Neural Networks: Theoretical Concepts, edited by V. Vemuri. IEEE Computer Society Press Technology Series, 1988.

Widrow B. and Smith F.W. (1964). Pattern-Recognizing Control Systems. in Computer and Information Science, edited by J.T. Tou and R.H. Wilcox, Spartan, 1964.

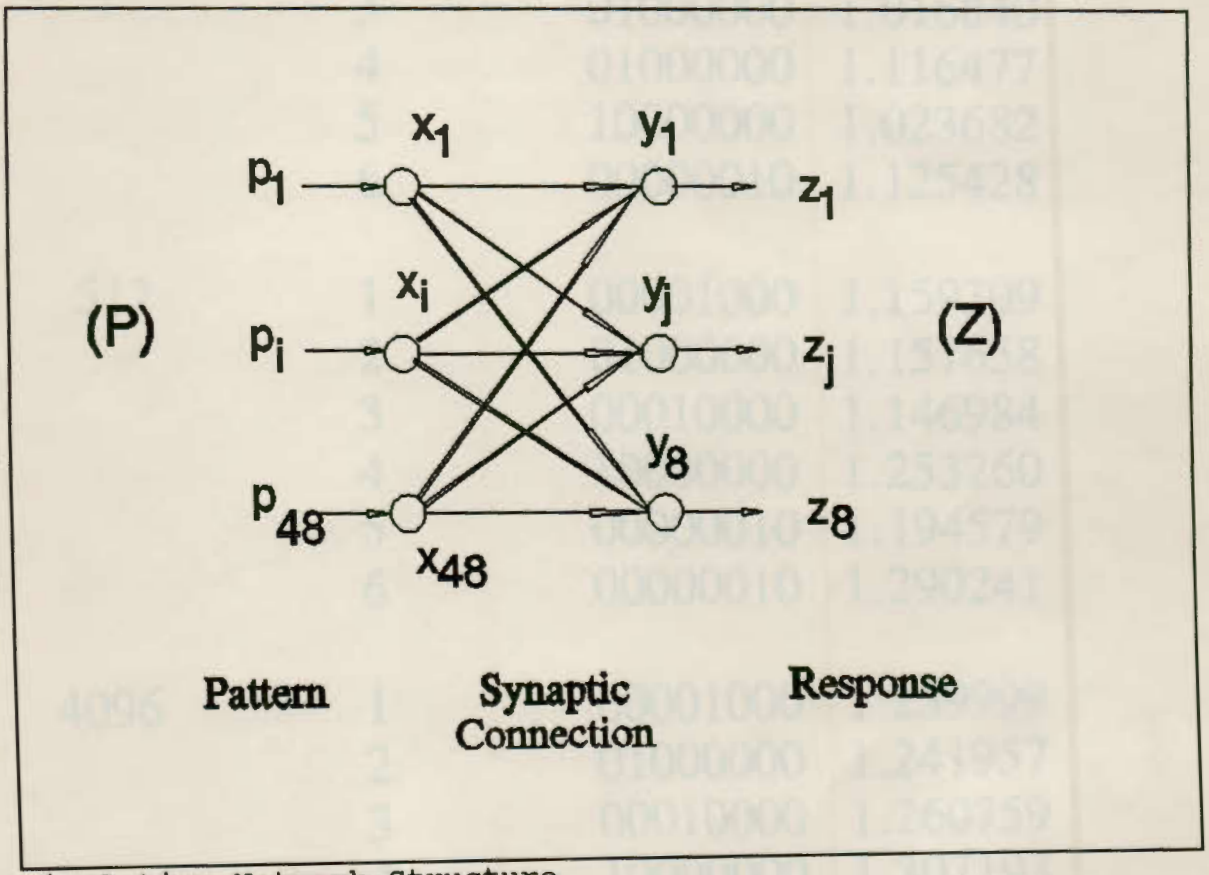
Widrow B., Gupta, Maitra, (1973). Punish/Reward: Learning with a Critic in Adaptive Threshold Systems. IEEE Transaction on System, Man and Cybernetics, SMC-3, 455-465, 1973.

Widrow B., McCool J.M., Larimore M.G. and Johnson C.R.,Jr. (1976). Stationary and Nonstationary Learning Charateristics of LMS Adaptive Filter. Proceedings of IEEE, Vol 64, NO 8, August 1976, 1151-1162.

Wyckoff B.L.Jr. (1954). A Mathematical Model and An Electronic Model for Learning. Psychology Review, 61, 89-97, 1954.

STRUCTURE OF NETWORK

Consider a neural network consisting of 48 neurons in the first layer and 8 neurons in the second layer. We set the constants q1 and q2 to be 25 and 1 respectively. The step size, γ, is 0.0001. The threshold, Θ_i, of each of the neurons in the first layer are set to 0. The update takes over 4096 iterations.



Simulation Network Structure

Test 1: General LAR approach

In this test, we choose pattern (i) to (vi).

Iteration	Pattern #	zj	max. value of yj
0	1	01000000	0.976141
	2	01000000	0.955053
	3	01000000	1.016840
	4	01000000	1.116477
	5	10000000	1.023682
	6	00000010	1.125428
512	1	00001000	1.159309
	2	01000000	1.157658
	3	00010000	1.146984
	4	10000000	1.253260
	5	00000010	1.194579
	6	00000010	1.290241
4096	1	00001000	1.259999
	2	01000000	1.241957
	3	00010000	1.260759
	4	10000000	1.307193
	5	00000010	1.271767
	6	00000001	1.320562

Test 2: Chaotic LAR approach

In this test, we select the pattern (i) to (iv) and (vii) to (viii) from Fig(4.2). The network parameters are the same. But here we add a concept - **Chaos** - to the simulation program.