

A STATISTICAL MODELING OF AN OLFACTORY
SYSTEM FOR ELECTRONIC
IMPLEMENTATION

By

RAHUL RATILAL SHAH

Bachelor of Engineering

University of Poona

Pune, India

1993

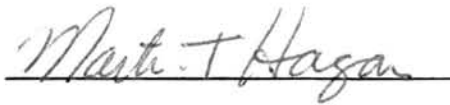
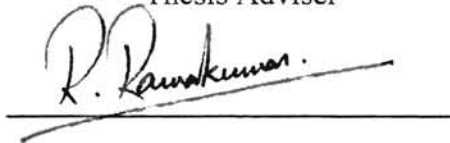
Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfilment of
the requirement for
the Degree of
MASTER OF SCIENCE
July, 1996

A STATISTICAL MODELING OF AN OLFACTORY
SYSTEM FOR ELECTRONIC
IMPLEMENTATION

Thesis approved:



Thesis Adviser



Dean of the Graduate College

PREFACE

This thesis attempts to provide an understanding of the clustering mechanism observed in the olfactory neural network. A new model is proposed that has potential to overcome the problems associated with the clustering mechanism. An additional layer of network that can extend the proposed network to LVQ network is also discussed.

I wish to express my sincere gratitude to my adviser Dr. C. Hutchens for his invaluable guidance, inspiration and financial support. I am also thankful to Dr. M. Hagan and Dr. R. Ramakumar for serving on my committee. I would like to thank Dr. P. Shoemaker for his guidance throughout this work.

I extend my thanks to my friends for their everlasting support and encouragement.

I express my deepest appreciation for my parents for their love, understanding and moral support.

Finally I would like to quote a couplet from ShreemadBhagavadGeeta that has inspired me at the moment of frustration:

Thy jurisdiction is in action alone

Never in its fruits at any time

Never should the fruits of action be thy motive

Never let there be attachment in thee to inaction

TABLE OF CONTENTS

Chapter	Page
1. ARTIFICIAL NEURAL NETWORKS AND OLFACTION.....	1
1.1 INTRODUCTION	1
1.2 BIOLOGICAL INSPIRATION.....	2
1.3 ARTIFICIAL NEURAL NETWORKS	3
1.4 OLFACTORY SYSTEM	5
1.5 REVIEW OF SOME IMPORTANT TOPICS	8
1.5.1 Hierarchical clustering mechanism.....	8
1.5.2 Associative Learning : Kohonen Network.....	9
1.5.3 Winner-Take-All (WTA) Learning Rule.....	10
1.5.4 Learning Vector Quantization (LVQ) Network.....	11
2. STATISTICAL MODEL OF OLFACTORY SYSTEM.....	14
2.1 OVERVIEW OF THE GLA MODEL.....	14
2.1.1 Block diagram and basic operation	14
2.1.2 Network description and nomenclature.....	17
2.2 OVERLAP AND HAMMING DISTANCE	17
2.3 PROBABILITY OF WINNING ON MULTIPLE PATTERNS IN THE NAIVE NETWORK	18
2.4 PROCESS OF CAPTURE.....	19
2.4.1 Capture by Precedence.....	20
2.4.2 Capture by Overtake.....	27
2.5 REVIEW OF THE COULTRIP-GRANGER MODEL	43
2.5.1 Network Description.....	43
2.5.2 Working of the Network.....	44
2.5.3 Advantages and disadvantages of the network.....	46
2.5.4 Similarity and difference with Kohonen network.....	47
2.6 PROPOSED MODEL	47
2.7 PROPOSED TRAINING ALGORITHM.....	48
2.8 ADVANTAGES OF THE PROPOSED MODEL OVER THE OLD MODEL.....	54
2.9 SIMULATIONS	55
2.9.1 Experiment 1.....	55
2.9.1 Experiment 2.....	57
2.9.3 Conclusion.....	58
2.10 RECONSTRUCTING THE TEMPLATE	59
2.11 LVQ NETWORK.....	66
3. CONCLUSIONS AND FUTURE PROSPECTS	70
REFERENCES	74

LIST OF FIGURES

Figure	Page No.
1 GLA MODEL : BLOCK DIAGRAM.....	15
2 P[SINGLE CELL WINS ON TWO PATTERNS (NAIVE NETWORK)].....	19
3 CAPTURE BY PRECEDENCE : N=960.....	22-23
4 CAPTURE BY PRECEDENCE : N=1920.....	23-24
5 CAPTURE BY PRECEDENCE : N=3840.....	25-26
6 CAPTURE BY OVERTAKE : N=960, m=2.....	29-30
7 CAPTURE BY OVERTAKE : N=1920, m=2.....	30-31
8 CAPTURE BY OVERTAKE : N=3840, m=2.....	32-33
9 CAPTURE BY OVERTAKE : N=960, m=4.....	33-34
10 CAPTURE BY OVERTAKE : N=1920, m=4.....	35-36
11 CAPTURE BY OVERTAKE : N=3840, m=4.....	36-37
12 CAPTURE BY OVERTAKE : N=960, m=8.....	38-39
13 CAPTURE BY OVERTAKE : N=1920, m=8.....	39-40
14 CAPTURE BY OVERTAKE : N=3840, m=8.....	41-42
15 COULTRIP-GRANGER MODEL : BLOCK DIAGRAM.....	44
16 PROPOSED MODEL : BLOCK DIAGRAM	48
17 TRAINING ALGORITHM : SIMPLIFIED FLOW CHART	52
18 TRAINING ALGORITHM : FLOW CHART	53
19 TRAINING ALGORITHM : TRAIN ROUTINE	54
20 EXPERIMENT 1	56
21 EXPERIMENT 2	57
22 BACKPROPAGATION THROUGH FIXED PATHWAY	60
23 LVQ LAYER : BLOCK DIAGRAM.....	67

NOMENCLATURE

g	number of mitral patches
n	number of mitral cells per mitral patch
N	number of LOT lines
r	LOT activity
ρ	overlap between two patterns
H	hamming distance between two patterns
ρ_{\min}	minimum overlap between the center of the cluster and vector in that cluster
P_n	P[single cell wins on two patterns having overlap ρ_{\min} between them]
P_{cp}	P[capture by precedence for ρ_{\min}]
W	weight matrix
Δw	weight increment
w_{sat}	saturated weight value
q	weight matrix sparsity
p	number of piriform patches
h	number of piriform cells per piriform patch
B	output of the binary encoder
v	size of the output of binary encoder
R	AND array in PLA
u	number of AND gates in PLA
A	outputs of the AND gates in PLA
O	OR array in PLA
Q	output of PLA

CHAPTER 1

ARTIFICIAL NEURAL NETWORKS AND OLFACTION

1.1 Introduction

From the ancient times, the goal of humankind has been to improve the quality of life. Different sciences and arts are the products of the effort of the humankind to achieve a goal that can never be fully achieved. Man has developed many machines such as digital computers to get rid of the tedious tasks and enjoy a more fruitful life. He/she is still trying to make these machines as fast and efficient as possible. Nowadays engineers and scientists are trying to develop intelligent machines. Artificial neural systems are present-day examples of such machines that have great potential to further improve the quality of our life.

The recent resurgence of interest in neural networks has its roots in the recognition that the brain performs computations in a different manner than do conventional digital computers. Although the digital computers outperform both biological and artificial neural systems for tasks based on precise and fast arithmetic

operations (rather the computers are meant for that), humans (or for that matter animals) are much more efficient and fault-tolerant than the fastest computers at computationally complex tasks such as pattern recognition. A neural network's ability to perform computations is based on the hope that we can reproduce some of the flexibility and power of the human brain by an artificial means.[1]

1.2 Biological Inspiration

Much of the inspiration for the artificial neural systems comes from biology, or to be more specific, from neuroscience. However, we are not directly concerned with the biological neurons or neural networks.

The brain consists of a large number (approximately 10^{11}) of highly connected elements (approximately 10^4 connections per element) called neurons. For our purposes, these neurons have three principal components: dendrites, cell body and axon. The dendrites (from Greek *dendron*, tree) are tree-like receptive networks of nerve fibers that carry electrical signals into the cell body. The cell body effectively sums and thresholds these incoming signals. The axon is a single long fiber which carries the signal from the cell body out to other neurons. The point of contact of an axon of one cell with a dendrite of another cell is called a synapse (from Greek *syn* + *haptien*, to copulate). It is the arrangement of neurons and the strengths of individual synapses, determined by complex chemical process, which determines the function of the neural network.[2]

Some of the neural structure is defined at birth. Other parts are developed through learning. Neural networks continue to adjust throughout the life. These later changes tend to consist principally of strengthening or weakening of synaptic junctions.

1.3 Artificial Neural Networks

Artificial neural networks (ANNs) are physical cellular systems which can acquire, store, and utilize experiential knowledge. The neurons that we consider in artificial neural networks are not biological. They are extremely simplified abstractions of biological neurons, realized as elements in programs or as circuits made of silicon. Networks of these artificial neurons do not approach the complexity of the brain. There are, however, two key similarities between biological and artificial neural networks. First the building blocks of both networks are simple computational devices (although artificial neurons are much simpler than biological neurons) which are highly interconnected. Secondly, the connections between the neurons determine the function of the network.

It should be noted that even though biological neurons are very slow when compared to electrical circuits (10^{-3} s compared to 10^{-9} s), the brain is able to perform many tasks much faster than any conventional computer. In contrast to a large software program which operates serially on a conventional computer, the brain operates with large number of parallelly operating functional paradigms that execute in a more efficient way. The brain is very resistant to noise and rather robust whereas most computers are fault intolerant [3]. ANNs share this parallel structure. Even though most ANNs are currently implemented on

conventional digital computers, their parallel structure makes them ideally suited to implementation using custom VLSI, optical devices or parallel processors.[2]

The functional model of an artificial neuron or processing element consists of weighted input connections, a summation function and a non linear threshold function which transforms the input to an output. In biologically inspired artificial neural networks, neurons are regular and offer highly interconnected topologies. ANNs are composed of many non-linear computational elements which operate in parallel and are arranged in a pattern reminiscent of biological neural network in which elements are connected via sparsely connected weights. Neural networks learn to recognize the patterns by adjusting the weights of connection between processing elements. In its simplest form, an electronic neuron sums weighted inputs and passes the result through a non-linearity. A neuron is characterized by an internal threshold or offset and by the type of non-linearity.

Neural networks are inherently adaptive in the manner in which they derive meaning from imprecise, ambiguous and faulty real world data. In an ANN one of the basic problem is how its interconnection strengths i.e. synapses adjust their weights so the resulting neural networks shows the modified properties of pattern recognition. That is what type of learning algorithm is adopted? Adaptation or learning is a major area of neural network research. An example of such adaptation is speech recognition where training data is limited. Neural network classifiers are non-parametric and they make weaker assumptions concerning the shapes of underlying distributions than traditional statistical classifiers do. Such neural network adaptive systems are often described by an energy function or probability distribution. Their adaptivity and non-linearity make neural networks best

suites to predict, control and optimize such processes as industrial inspection, control, robotics, market timing, stock picking and credit approval.

ANNs are very effective in processing complex scientific and engineering problems such as speech processing, pattern mapping, pattern recognition and image recognition. Neural network models exhibit a number of properties analogous to the brain, e.g., generalization, parallel search, learning and flexibility.

Adaptation in neural networks also provides a degree of robustness by compensating for minor variability in characteristics of processing elements as well as sparse inputs. Most sequential computers are less fault intolerant when compared to the neural networks which are resistant to noise and robust, meaning that a fault to an individual neuron or even a group of neurons does not degrade the overall performance of the brain whereas the loss in one memory location of classical computer memory results in complete system failure.[3]

1.4 Olfactory System

In biological neural networks, the modulation of synaptic strength has long been regarded as the likely mechanism for learning and memory [4]. The long term potentiation (LTP) that is observed in the hippocampus, limbic system, and other cortical structures of the brain uses a similar mechanism for learning [5]. The synaptic strength due to LTP changes rather coarsely compared to graded and precise weight changes required by the artificial neural networks such as back propagation or Kohonen's self learning networks. How a nervous system might work within such constraints to perform useful computation

and to learn effectively is an elusive question [6]. Understanding the multiple ways in which the brain uses neuronal system for pattern processing serves as a key to the design of a neural network model. Elucidation of the sensory code by which information is transduced into neural information has been the major goal of investigators for several decades.

Of all the sensory modalities, the olfactory (from Latin *olefacere*, to smell) system is one of the least understood in terms of anatomy and physiology as well as the least studied for electronic modeling. What might be the simplest possible olfactory neural network model that is capable of learning and recognizing odors is a question. R. Granger, G. Lynch, and Ambros-Ingerson of U. C. Irvine have reported a potentially useful model for the investigation of the aggregate network learning and memory properties of olfaction in behaving animals. This model referred to as the GLA model henceforth, deals with the interacting structures of the olfactory bulb and piriform cortex that has been observed in rats [7,8,9]. Computer simulations of this model have been reported to have attractive computational properties, such as,

- the ability to identify clusters in the input cue environments at various levels of detail, hence achieving a form of hierarchical clustering
- the extensibility to unsupervised learning
- the ability to detect weak odor obscured by a stronger one or identifying a the component of a complex odor.

A central feature of this model is the periodic sampling of input odors at the theta rhythm to which the network response is locked. The theta rhythm matches the rhythm for

both the hippocampal firing and the rate at which rats sample odors during learning. With successive sniffs of inputs, hierarchical clustering and unmasking operations have been reported to proceed sequentially. On the first sniff, a general class, let us call it as “*parent class*”, is identified. On the second sniff, a class (child class), derived from the parent class, is identified. The child class inherits the properties of the parent class and has its own additional features. On the third sniff, some child class, derived from the child class discussed above, is identified and so on. In other words, on the first sniff, an abstract class is identified. All the classes identified in the successive sniffs hang down to this class.

The GLA olfactory model can be nicely distinguished from traditional abstract neural networks as the olfactory model more closely imitates the nervous systems. The GLA olfactory model (a tightly coupled network) employs sparsely connected networks of more elaborate neurons in which substantial information processing occurs within a single neuron whereas most ANN are typically comprised of densely connected layered network of simple neurons [4].

The GLA olfactory model shows great promise for the classification of unknown signal vectors by hierarchical clustering. The clustering of the input signals takes place by a method referred to as “process of capture.” However the capture mechanism in the GLA olfactory model has some problems associated with it. R. Coultrip and R. Granger have discussed a new model for sparse random networks with LTP learning rules that can have the problems associated with the GLA model partially solved. This model is statistically similar to the Kohonen network. The Coultrip-Granger model is reviewed in Chapter 2. A new model, which is a modified version of the Coultrip-Granger model, is

presented. The training algorithm for this model and the errors associated with the reconstruction of the template are discussed at length. It is shown that the newly proposed model is statistically similar to the Kohonen network. Further, for the proposed model, it is shown that if an additional layer is added to the network, it can be converted to the LVQ (learning vector quantization) network. The objective of this effort is to validate the model by simulations and discuss the additional linear layer to convert the model to LVQ while maintaining biological plausibility, quicker learning and electronic implementability

1.5 Review of some important topics

Before proceeding to Chapter 2 let us review some of the important topics in neural networks which are used in the rest of the thesis report.

1.5.1 Hierarchical clustering mechanism

In a hierarchical classification the data is not partitioned into a particular number of classes or clusters at a single step. Instead the classification consists of a series of partitions which may run from a single cluster containing all individuals, to n clusters each containing a single individual. Hierarchical clustering techniques may be subdivided into *agglomerative methods* which proceed by a series of successive fusions of the n individuals into groups, and *divisive methods*, which separate the n individuals successively into finer groupings. Both types of hierarchical clusterings can be viewed as

attempting to find the most efficient step, in some defined sense, at each stage in the progressive subdivision or synthesis of the data. With such methods, divisions or fusions once made are irrevocable, so that when an agglomerative method has joined two individuals, they cannot subsequently be separated and when a divisive algorithm has made a split, it cannot be reunited. In other word, the hierarchical clustering suffers from the defect that it can never repair what was done in previous steps. Since all agglomerative hierarchical techniques ultimately reduce the data to a single cluster containing all the individuals, and the divisive techniques will finally split the entire set of data into n groups each containing a single individual, the investigator wishing to have a solution with an *optimal* number of clusters, will need to decide on a particular stage to stop.[11]

1.5.2 Associative Learning : Kohonen Network

When a particular pattern (*stimulus*) is applied to a network, it is trained to produce some particular pattern as *response*. This is called associative learning. Association between the input and output patterns of the network is the fundamental aspect of associative memory. There are several algorithms to implement associative learning. Let's review the Kohonen's rule for associative learning which can be stated as follows:

$$\mathbf{w}_i^{new} = \mathbf{w}_i^{old} + \alpha [\mathbf{p}(q) - \mathbf{w}_i^{old}] \quad \text{for } i \in X(q) \quad (1.1)$$

where α is the learning rate, \mathbf{w}_i is the i th column of the weight matrix, $\mathbf{p}(q)$ is the q th input pattern (stimulus). It can be seen from the equation, Kohonen's rule tries to make the i th column of the weight matrix as similar to the stimulus as possible.[2]

1.5.3 Winner-Take-All (WTA) Learning Rule

This rule is an example of Competitive learning, and it is used for unsupervised network training. Typically winner-take-all learning is used for learning statistical properties of inputs. The learning is based on the premise that one of the neurons in the layer, say the i th, has the maximum response due to input \mathbf{p} . This neuron is declared the *winner*.

Let \mathbf{w} be the weight matrix ($m \times n$). As we have assumed, let i th neuron be the winner. As a result of this winning event, the weight vector \mathbf{w}_i

$$\mathbf{w}_i = [w_{1i} \ w_{2i} \ w_{3i} \ \dots \ w_{ni}]^T$$

is the only one adjusted in the unsupervised learning step. The learning rule can be stated as follows

$$\begin{aligned} \mathbf{w}_j^{new} &= \mathbf{w}_j^{old} + \alpha [\mathbf{p} - \mathbf{w}_j^{old}] & \text{for } j = i \\ &= \mathbf{w}_j^{old} & \text{for } j \neq i \end{aligned} \quad (1.2)$$

where $j = 1, 2, 3, \dots, m$ and α is learning rate, typically decreasing as learning progresses. The winner selection is based on the following criterion of maximum activation among all m neurons participating in the competition.

$$\mathbf{w}_i^T \mathbf{p} = \max_{j=1,2,\dots,m} (\mathbf{w}_j^T \mathbf{p}) \quad (1.3)$$

This criterion corresponds to finding the weight vector that is closest to the input \mathbf{p} . Thus the rule (1.2) reduces to incrementing \mathbf{w}_i by a fraction of $\mathbf{p} - \mathbf{w}_i$. After the weight adjustment, the weight vector corresponding to the winner neuron tends to be better estimate of the input pattern in question. Weights are typically initialized at random values and their values are normalized during learning in this method.[2]

1.5.4 Learning Vector Quantization (LVQ) Network

The LVQ network is a hybrid network which uses both unsupervised and supervised learning to form classifications. It is a two layer network. The first layer of the LVQ network is a competitive layer and the second layer is a linear layer.

As with the competitive network, each neuron in the first layer learns a prototype vector which allows it to classify a region of the input space. In the competitive network, the neuron with the nonzero output indicates which class the input vector belongs to. For the LVQ network, the winning neuron indicates a *subclass*. There may be several different neurons (subclasses) which make up each class.

The second layer of the LVQ network is used to combine subclasses into a single class. Let the weight matrix associated with the second layer be \mathbf{w}^2 . The columns of \mathbf{w}^2 represent subclasses, and the rows represent classes. \mathbf{w}^2 has a single 1 in each column, with the other elements set to zero. The row in which the 1 occurs indicates which class the appropriate subclass belongs to.

$$(w_{ki}^2 = 1) \Rightarrow \text{subclass } i \text{ is a part of class } k$$

The process of combining subclasses to form a class allows the LVQ network to create complex class boundaries. A competitive layer alone has the limitation that it can only create decision regions which are convex. But the actual clusters can be concave.[2]

LVQ Network Learning with the Kohonen Rule

The learning in the LVQ network combines competitive learning with supervision. As with all supervised learning algorithms, it requires a set of examples of proper network behavior:

$$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2, \mathbf{t}_2\}, \dots, \{\mathbf{p}_n, \mathbf{t}_n\}.$$

Before learning can occur, each neuron in the second layer is assigned to an output neuron. This generates the matrix \mathbf{w}^2 . Typically equal number of neurons are connected to each output neuron, so that each class can be made up of some number of convex regions.

Once \mathbf{w}^2 is defined, it will never be altered. Let \mathbf{a}^1 be the output of the competitive layer and \mathbf{a}^2 be the output of the linear layer i.e. the final output. The weight matrix associated with the competitive layer (\mathbf{w}^1) is trained with a variation of Kohonen rule. The output of the winning neuron, say i th neuron, is set to 1 whereas the outputs of the remaining neurons are reset to zero. Next \mathbf{a}_i is multiplied by \mathbf{w}^2 to get \mathbf{a}^2 which also has only one non-zero element, say k th element, indicating that \mathbf{p} belongs to class k .

Kohonen's rule is used to improve the performance of the competitive layer of the network. If the pattern \mathbf{p} is classified correctly, then the weights of the winning neuron are moved toward \mathbf{p} .

$$\mathbf{w}_i^{new} = \mathbf{w}_i^{old} + \alpha [\mathbf{p}(q) - \mathbf{w}_i^{old}], \quad \text{if } a_k^l = t_k = 1 \quad (1.4a)$$

If \mathbf{p} is classified incorrectly, then the weights of the winning neuron are moved away from \mathbf{p} .

$$\mathbf{w}_i^{new} = \mathbf{w}_i^{old} - \alpha [\mathbf{p}(q) - \mathbf{w}_i^{old}], \quad \text{if } a_k^l = 1 \neq t_k = 0 \quad (1.4b)$$

Equation (1.4) is the modified Kohonen Rule for LVQ networks. As a result of this learning process, each neuron moves towards vectors that fall into the class for which it forms the subclass, and away from vectors which fall into other classes.[2]

CHAPTER 2

STATISTICAL MODEL OF OLFACTORY SYSTEM

Before proceeding to the discussion of the clustering mechanism in the olfactory system, it is worth taking an overview of the original GLA model.

2.1 Overview of the GLA model

2.1.1 Block diagram and basic operation:

A simplified block diagram of the GLA model is shown in Fig. 1.

The olfactory bulb receives excitatory input from the olfactory receptors in a roughly topographic fashion: receptor cells of a particular type project axons to a delimited area of the bulb termed as a glomerulus (from Latin *glomer*, ball). The aggregate firing rate of receptors of each type is regarded as a component of an input vector to the system. Each glomerulus is also subject to specific inhibitory feedback from the piriform cortex and the total activity in the bulb is mediated by lateral inhibitory cells which perform an effective normalization. The excitatory (“mitral”) cells in the glomeruli

are regarded as two-state devices (either active or inactive) and they have differing thresholds of excitation. The output of the glomeruli are thermometer-coded versions of the inhibited/normalized inputs, with signal intensity represented by the number of active cells in the glomerulus. The normalization constrains the bulb so that only some fraction (assumed to be 0.2) of the mitral cells can become active. The outputs of the mitral cells project to the piriform cortex via the lateral olfactory tract (LOT).

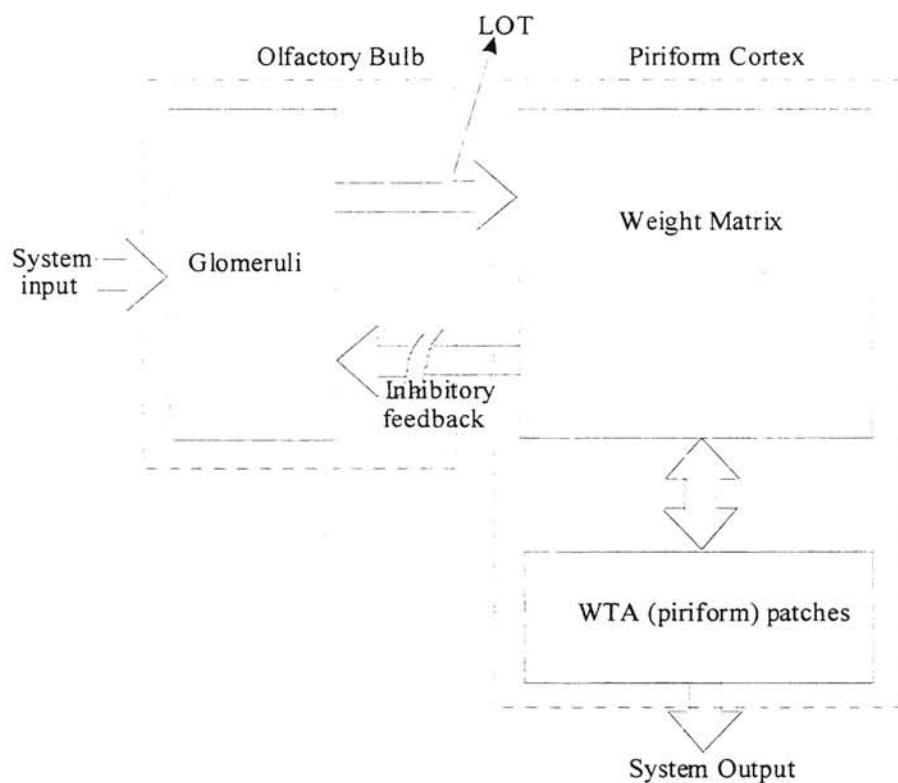


Fig. 1 GLA Model : Block Diagram

The LOT forms excitatory synapses with cells in the piriform in a sparse random fashion: connections appear to be made essentially at random with low probability (assumed to be 0.1). Piriform cells are also two-state devices and are arranged winner-

take-all patches in which lateral inhibition insures that only one cell at a time becomes active. The sparse patterns of piriform activity is the system output.

After stimulation, active piriform cells in turn inhibit the glomeruli in the bulb via a feedback pathway. Connections are developed according to a correlational learning rule so that the glomeruli most responsible for the firing of a piriform cell are inhibited most strongly. Feedforward excitation of the bulb followed by feedback inhibition of the piriform is repeated at the so-called theta rhythm, to which operation of the system is synchronized. After the first sniff of the odor, the most active glomeruli are inhibited the most strongly, allowing expression of secondary components of the input vector on the next sniff, and so on in a hierarchical fashion during subsequent sniffs.

The scope of this thesis is restricted to the open loop (i.e. feedforward) system.

Learning in the model is based on coactivity: the weights of synapses from active mitral onto winning piriform cells are incremented when learning is enabled. This learning rule is called long term potentiation or LTP. Weights saturate, and when fully potentiated are two or three times larger than the naive weights. Increments are coarse, comprising 10%-20% of the range between naive and saturated weights. There is no weight decay. The net effect of learning is that the network tends to cluster its inputs: the output codes for sufficiently similar input vectors become very similar or identical. Moreover, the feedback inhibition of the bulb by the piriform tends to inhibit the glomeruli not in proportion to their activity, but rather in relation to the expected activity of the cluster mean. The net result is that a hierarchical clustering takes place during the

multisampling (sniffing) process, in which the initial codes indicate the broad class membership and subsequent codes, subcluster or narrower class membership.[12]

2.1.2 Network description and nomenclature:

There are g inputs to the system. Corresponding to each input there is one mitral patch consisting of n mitral cells (i.e. n levels in the thermometer code). Size of the output of the olfactory bulb (LOT vector) is $ng \times 1$. Let us denote the product ng by N . The activity of the LOT vector (approximately 0.2) is denoted by r .

There are p piriform (WTA) patches. Each patch consists of h piriform cells in a piriform patch. So the size of the system output will be $1 \times hp$.

The weight matrix is denoted by W and its size is $N \times hp$. The weight matrix sparsity (approximately 0.1) is denoted by q . The weight increments are denoted by Δw . The naive (untrained) weights are assumed to have a value of 1. The maximum value a weight can attain is denoted by w_{sat} . For simulations Δw and w_{sat} are assumed to be 0.4 and 3 respectively.

2.2 Overlap and Hamming distance

Overlap between two patterns (LOT vectors) is defined as the fraction of the total number of active lines in any pattern active for both the patterns. It is denoted by ρ .

Hamming distance between two binary strings is the total number of mismatching bits. It can be given by the number of the non-zero bits in the binary string obtained by XORing the two binary strings in question. The Hamming distance is denoted by H .

It can be shown that $H=2Nr(1-\rho)$.

2.3 Probability of winning on multiple patterns in the naive network

Let us assume that a pattern P_1 is presented to the network and cell C_1 from a particular patch wins. Let us further assume that the network is not yet trained. Now a pattern P_2 is presented to the network. There exists a finite probability that the same cell C_1 wins on pattern P_2 also. This probability is a function of overlap ρ between patterns P_1 and P_2 and the piriform patch size h .

By intuition, for $\rho=0$, the probability should be $1/h$ since patterns P_1 and P_2 will be totally different and all h cells will have equal probability to win. For $\rho=1$, obviously the probability is 1 since the same pattern is presented again.

The theoretical expression for P [single cell wins on two patterns] are derived in [12]. Simulations results agree with the theory. Fig. 2 shows the probability curves for different values of patch-size h . It can be seen that this probability is a function of overlap ρ and patch size h . The figure suggests that the piriform patch should have a moderate number of piriform cells. Because for very small overlap also, there is certain probability that one cell wins on both the patterns. But as h increases, this probability decreases.

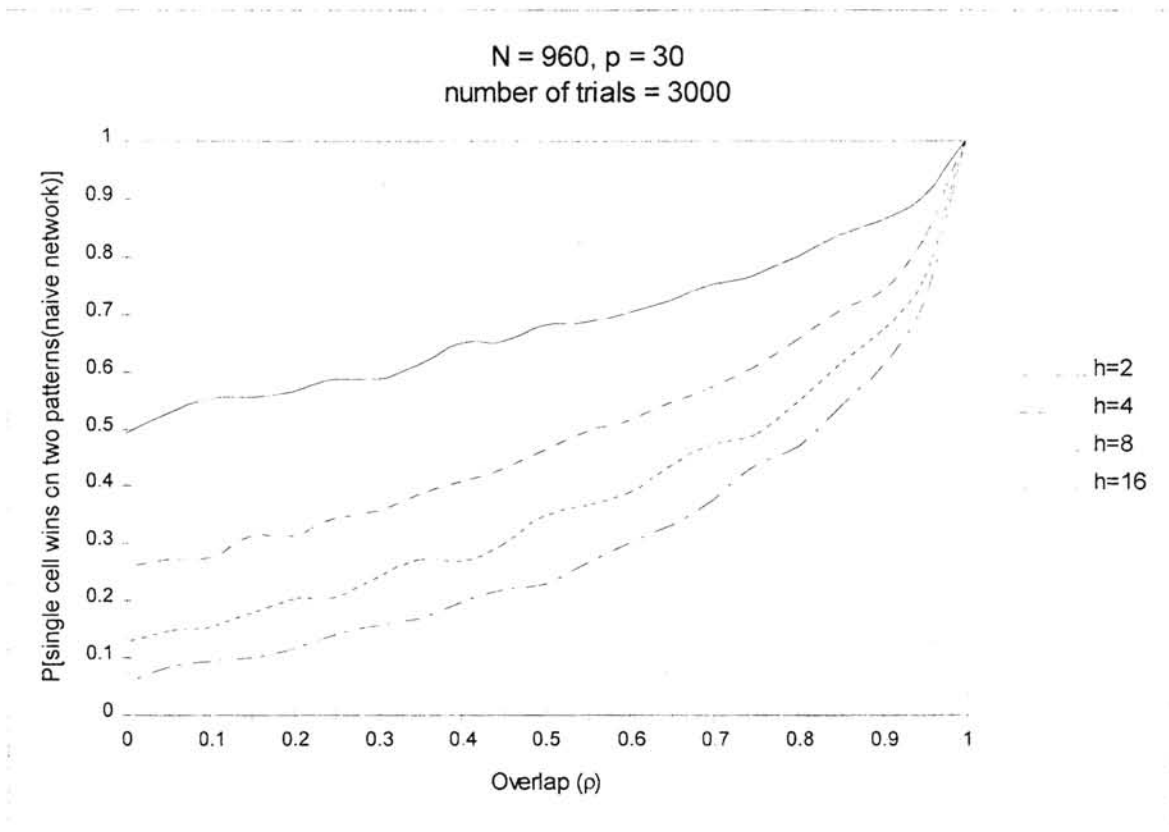


Fig. 2 P[single cell wins on two patterns (naive network)]

2.4 Process of Capture

As the GLA network starts learning, it acquires the ability to cluster the input data, i.e., for similar inputs, the output codes become more overlapping although they are not necessarily so in the naive network. This indicates that some cells **capture** patterns in the cluster as learning progresses at the loss of other cells which won in the naive network. Two types of the capture mechanisms have been identified and are defined below. [12]

1. Capture by precedence;

2. Capture by overtake.

2.4.1 Capture by Precedence

Capture by precedence is defined as the capture by virtue of precedence in the order in which the cells win during training.

Let us assume that in a particular patch, a cell, C_1 , wins on a number of patterns before some pattern, P , is presented to the network which would elicit the winner from some other cell, C_2 , in the naive network. But the network is already trained on a number of patterns on which the cell C_1 wins. When P is presented to the network, if the coactivation of C_1 exceeds the coactivation of C_2 , C_1 will be the winner even if pattern P is not very similar to the patterns on which C_1 won.

Capture by precedence is desired to some extent since it gives the network the ability to cluster the input signals. However, this can create problems when it starts capturing the patterns which are not members of the same cluster.

Simulations were carried out for different sets of parameters, i.e. different values of N, h . The general procedure is as follows:

1. Generate a random sparse weight matrix with sparsity $q=0.1$.
2. Generate a random LOT vector with activity $r=0.2$ and present to the network.
Determine the winners. Train the network.
3. Generate a series of vectors with overlaps ranging from 0 to 1 with the vector generated in (2) and present to the naive network first. Determine the winners.

4. Compare the winners of (2) and (3). The number of different winners is the number of “capturable” cells.
5. Present the vectors generated in (3) to the trained network. Determine the winners.
6. Compare the winners of (3) and (6). The number of different winners will give the number of captured cells.
7. $P[\text{capture by precedence}] = (\text{number of captured cells})/(\text{number of “capturable” cells})$
8. Repeat (1)-(7) several times and determine the average of the results.

The simulation results are shown in Fig. 3, 4 and 5.

If we see the figures 3.1, 4.1 and 5.1, it can be seen that even for a very small overlap (except 0) there are some cells captured by precedence. This number is a function of overlap ρ , patch size h , as well as the size of LOT vector, N . As N increases (from figure 3.1 to 5.1) the number of captured cells also increases. This fact is also reflected in the probability curves shown in figures 3.2, 4.2 and 5.2 which show that the probability of capture by precedence increases as N increases.

It can be seen that $P[\text{capture by precedence}]$ increases as N . It can be shown that a cell which wins on even a single pattern, if its weights are first ones to be incremented, will capture any other pattern with non-zero overlap of the first, provided N is sufficiently large.[12] This restricts the size of the LOT vector. We cannot increase the LOT size indefinitely.

Figures 3.3, 4.3 and 5.3 show the probability curves for a single cell winning on two patterns in the trained network. These curves show that as N increases (from figure 3.3 to 5.3) these curves shift upward.

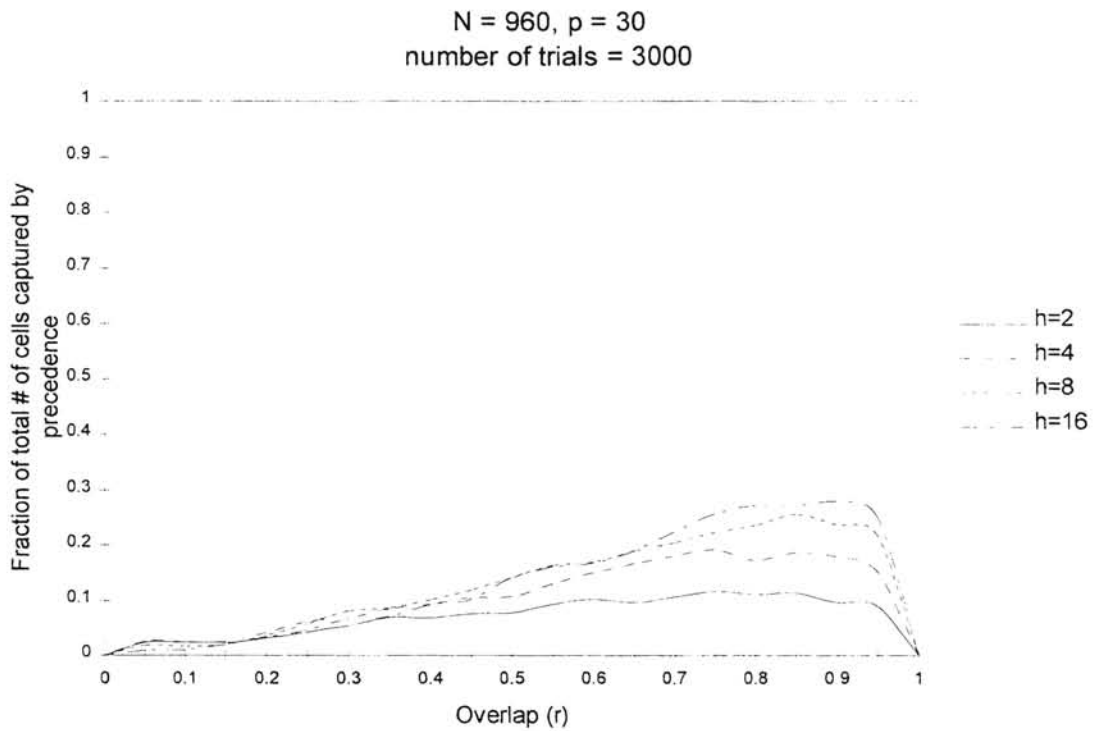


Fig 3.1 Capture by Precedence : N = 960

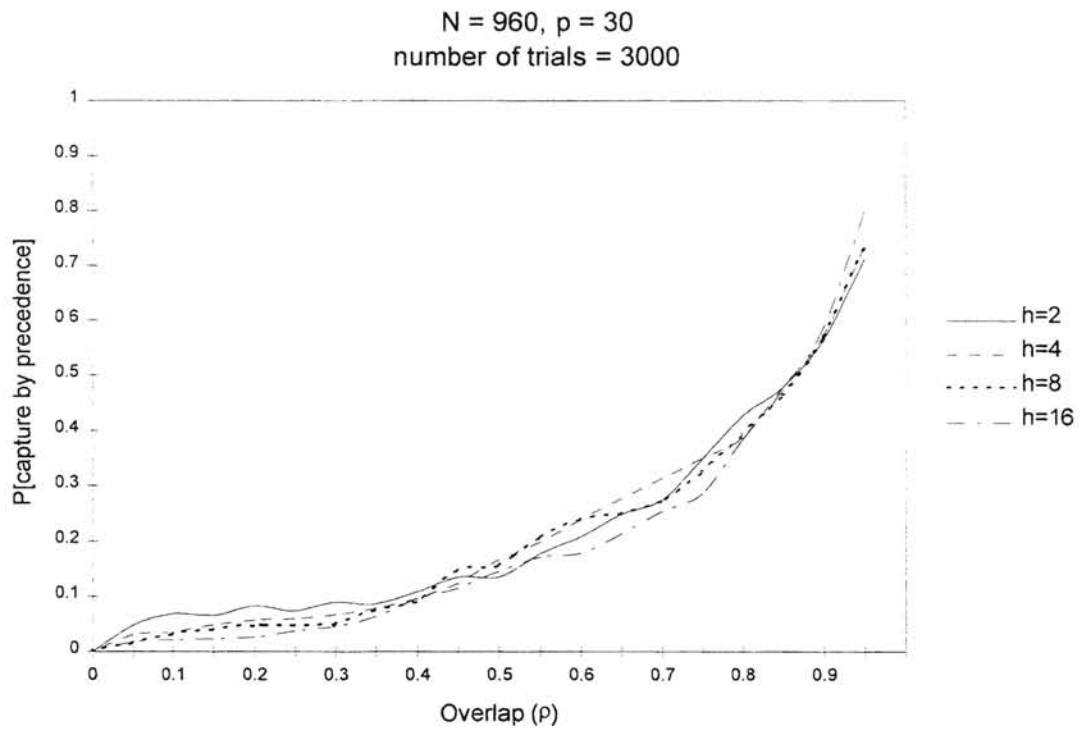


Fig 3.2 Capture by Precedence : N = 960

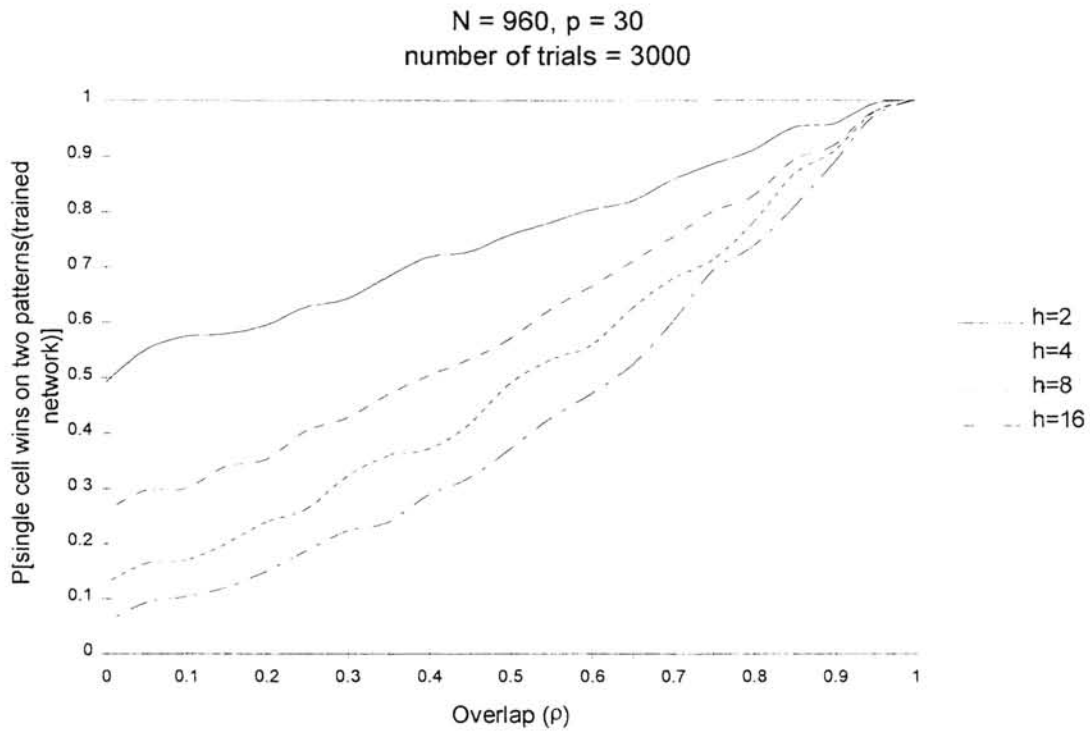


Fig 3.3 Capture by Precedence : N = 960

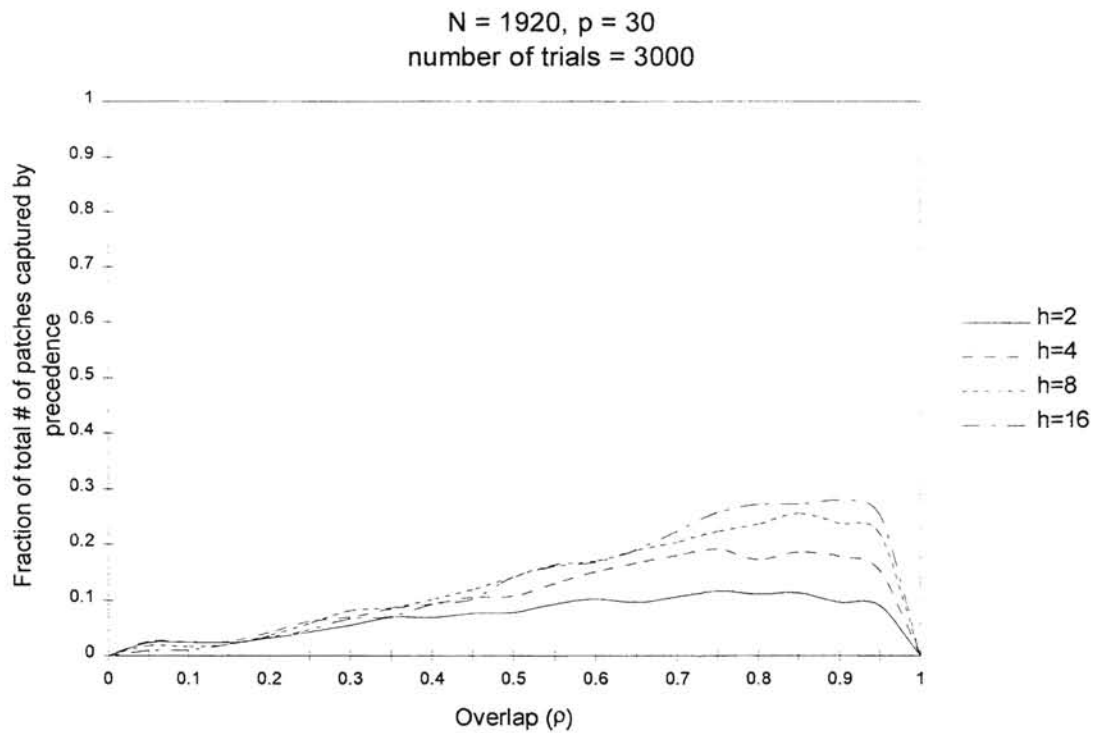


Fig. 4.1 Capture by Precedence : N = 1920

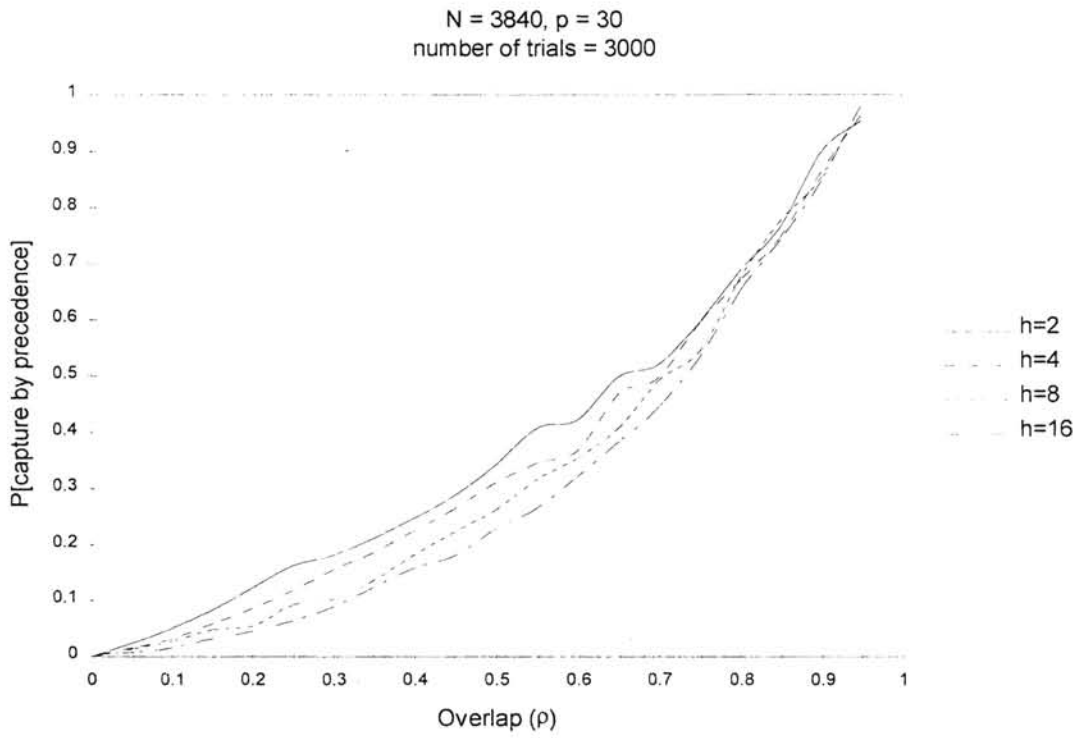


Fig 4.2 Capture by Precedence : N = 1920

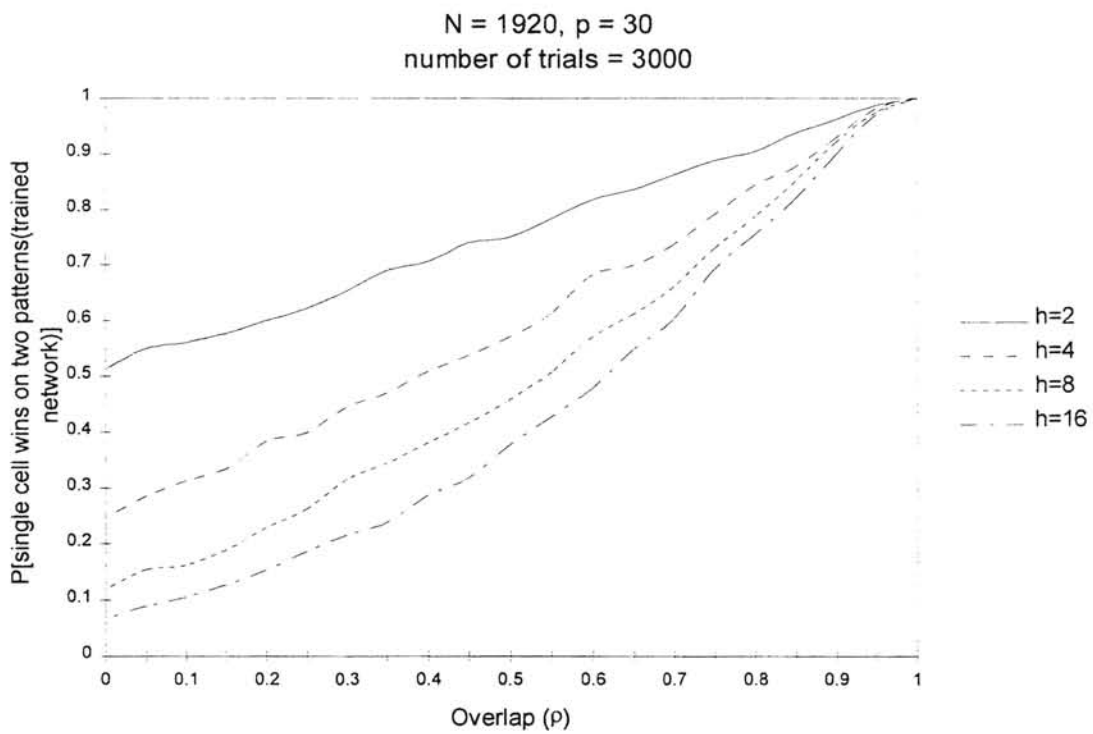


Fig 4.3 Capture by Precedence : N = 1920

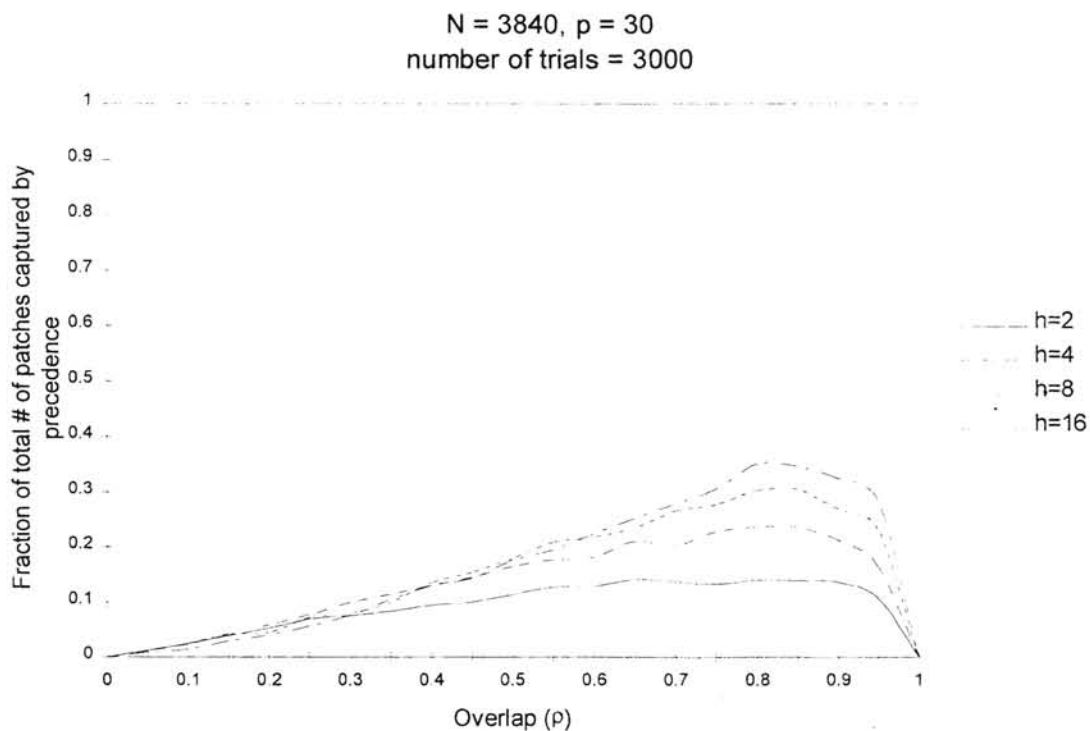


Fig 5.1 Capture by Precedence : N = 3840

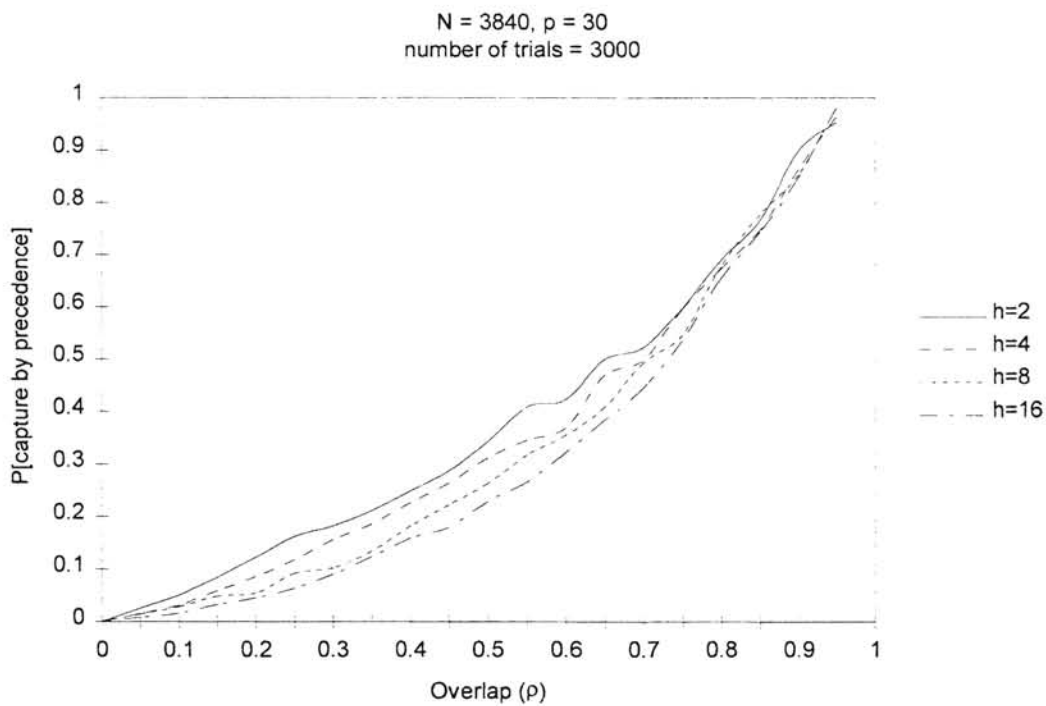


Fig 5.2 Capture by Precedence : N = 3840

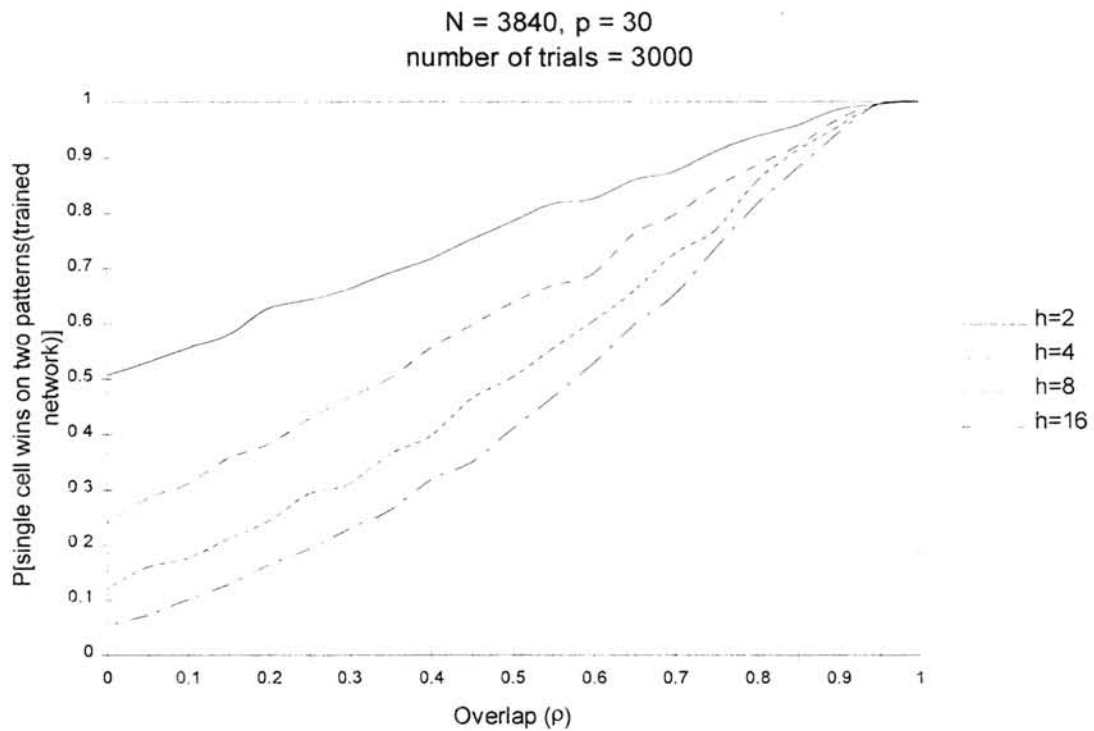


Fig 5.3 Capture by Precedence : N = 3840

To summarize:

- even for a very small overlap (except 0) between the input patterns, there exists a finite probability of capture by precedence.
- capture by precedence is dependent on the size of the input vector (N)
- it is also dependent on patch-size h

We will be using these figures in the training algorithm for the proposed model.

2.4.2 Capture by Overtake

Let pattern P_1 elicit a winner from cell C_1 and patterns P_k ($k = 2, 3, \dots, m+1$) elicit winners from cell C_2 . Let's further assume that P_1 and P_k each appear once per training epoch during learning and there is no capture by precedence during first training epoch. For cell C_2 to capture pattern P_1 , the growth rate due to the input to C_2 for P_1 must exceed the growth rate of the input to C_1 for P_1 during training. Capture by this mechanism is defined as **capture by overtake**.

Capture by overtake is desired to some extent as it gives the ability to the network to capture the patterns that were not classified correctly during the first training epoch. But it can be undesired if it allows capture of a pattern which is not very similar to the patterns eliciting winners from the capturing cell. Capture by overtake can also become more problematic if the training data is not fair, i.e. some clusters are overrepresented whereas others are underrepresented.

Simulations were carried out for different sets of parameters, i.e. different values of N, h, m . The general procedure is as follows:

1. Generate a random sparse weight matrix with sparsity $q=0.1$.
2. Generate a random LOT vector with activity $r=0.2$ and present to the network.

Determine the winners. Train the network m times.

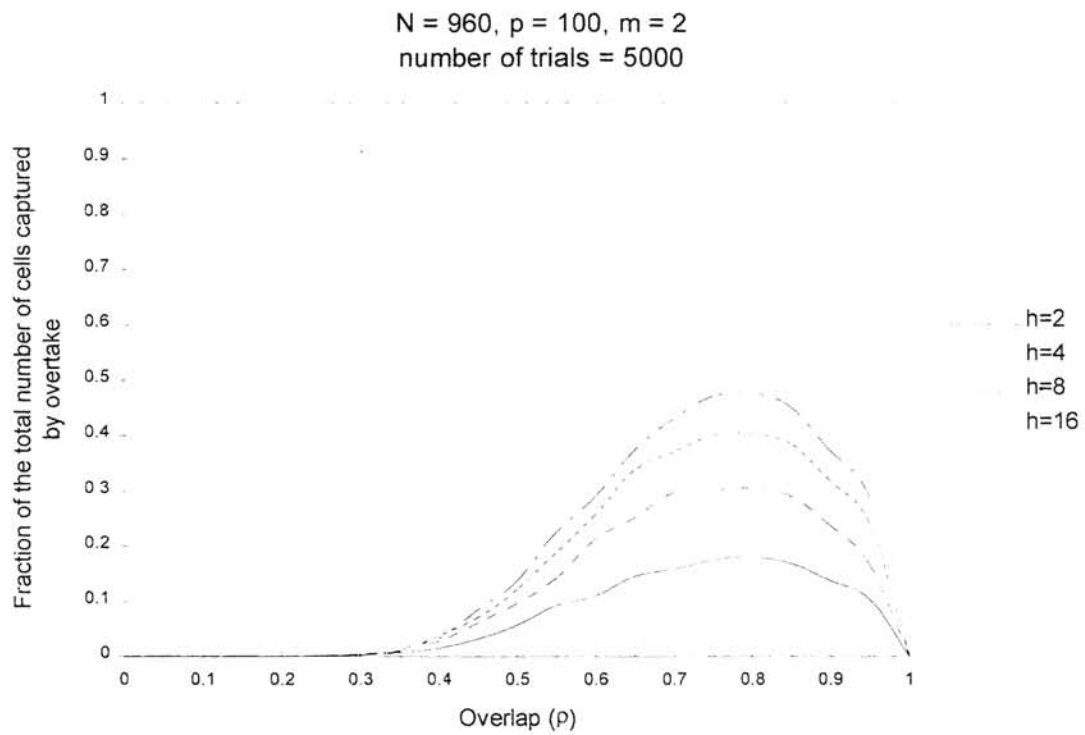
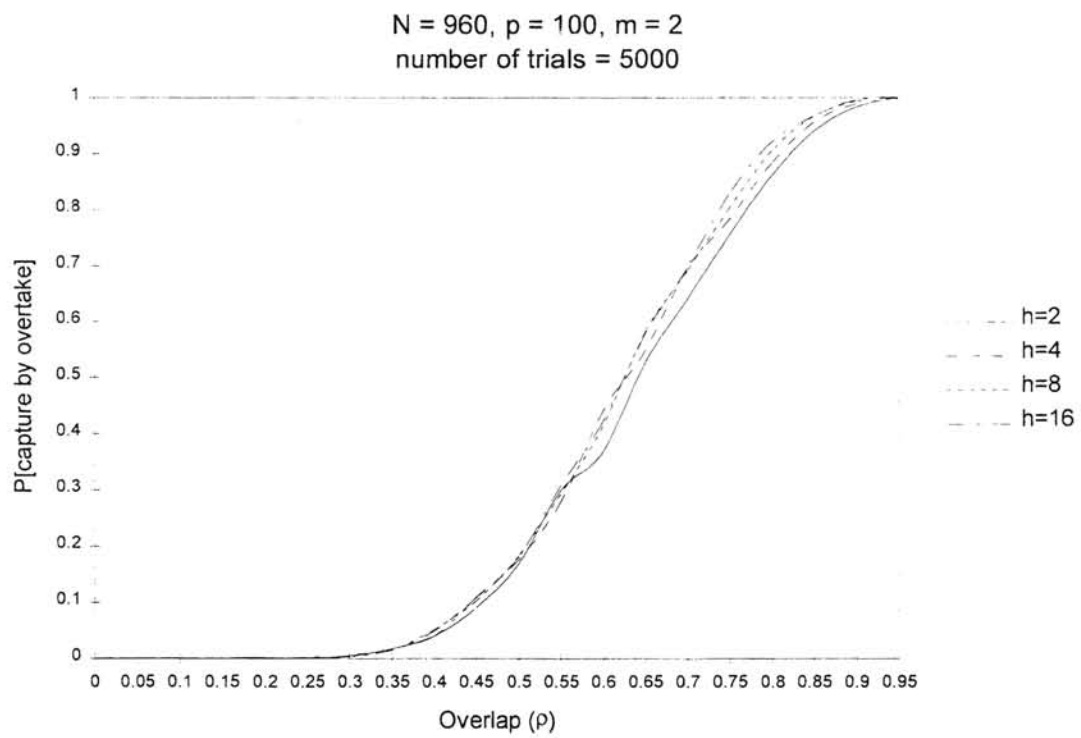
3. Generate a series of vectors with overlaps ranging from 0 to 1 with the vector generated in (2) and present to the naive network first. Determine the winners.

4. Compare the winners of (2) and (3). The number of different winners is the number of “capturable” cells.
5. Present the vectors generated in (3) to the trained network. Determine the winners.
6. Compare the winners of (3) and (6). The number of different winners will give the number of captured cells.
7. $P[\text{capture by overtake}] = (\text{number of captured cells})/(\text{number of “capturable” cells})$
8. Repeat (1)-(7) several times and determine the average of the results.

The simulation results are shown in figures 5-13.

If we compare figures 6.1, 9.1 and 12.1 (N constant, m is varied), it can be seen that as m increases more and more cells are captured for all values of overlap ρ (except 0 and 1). For $m=1$, these curves should be identical to capture by precedence curves. It shows that overrepresentation of some cluster can create error in classification of the patterns in other clusters.

If we compare figures 6.1, 7.1 and 8.1 (m constant, N increases), it can be seen that as N increases, a fewer number of cells are captured by overtake for overlap $\rho < 1/m$ and more number of cells are captured for overlap $\rho > 1/m$. As N increases the $P[\text{capture by overtake}]$ curves (figures 6.2, 7.2, 8.2) become more and more sharp. This is reflected in the $P[\text{single cell wins on 2 patterns (trained network)}]$ curves (fig. 7.3, 8.3, 9.3) also.

Fig 6.1 Capture by Overtake : $N = 960, m = 2$ Fig 6.2 Capture by Overtake : $N = 960, m = 2$

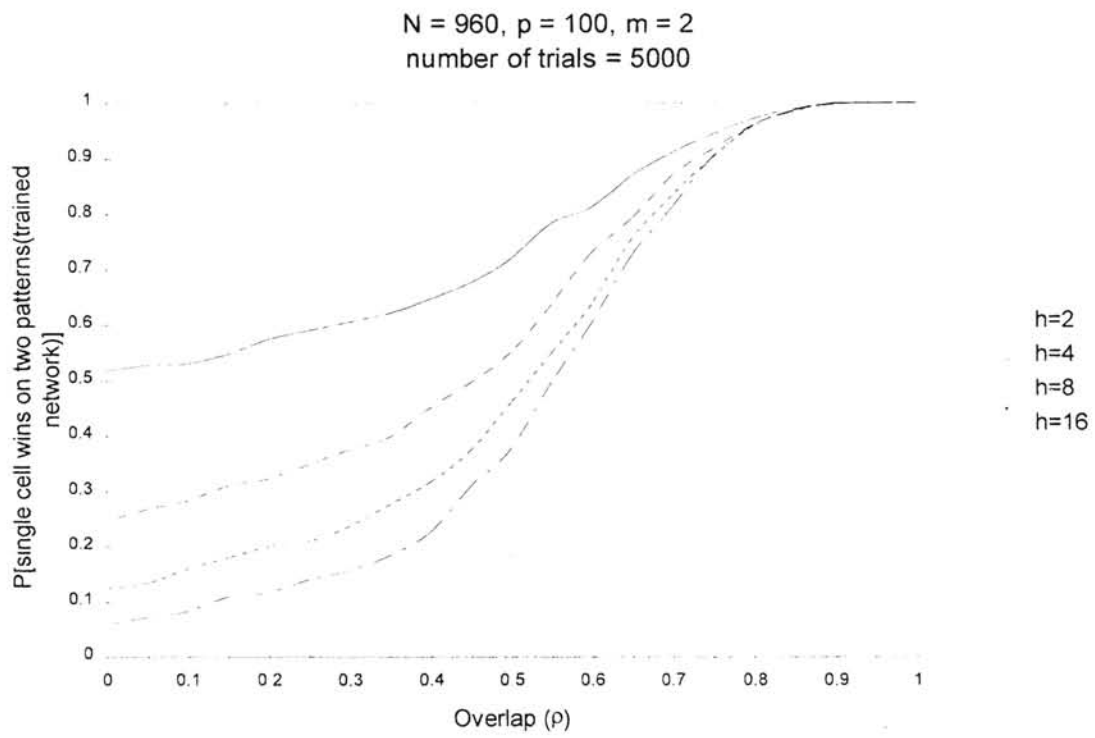


Fig 6.3 Capture by Overtake : $N = 960, m = 2$

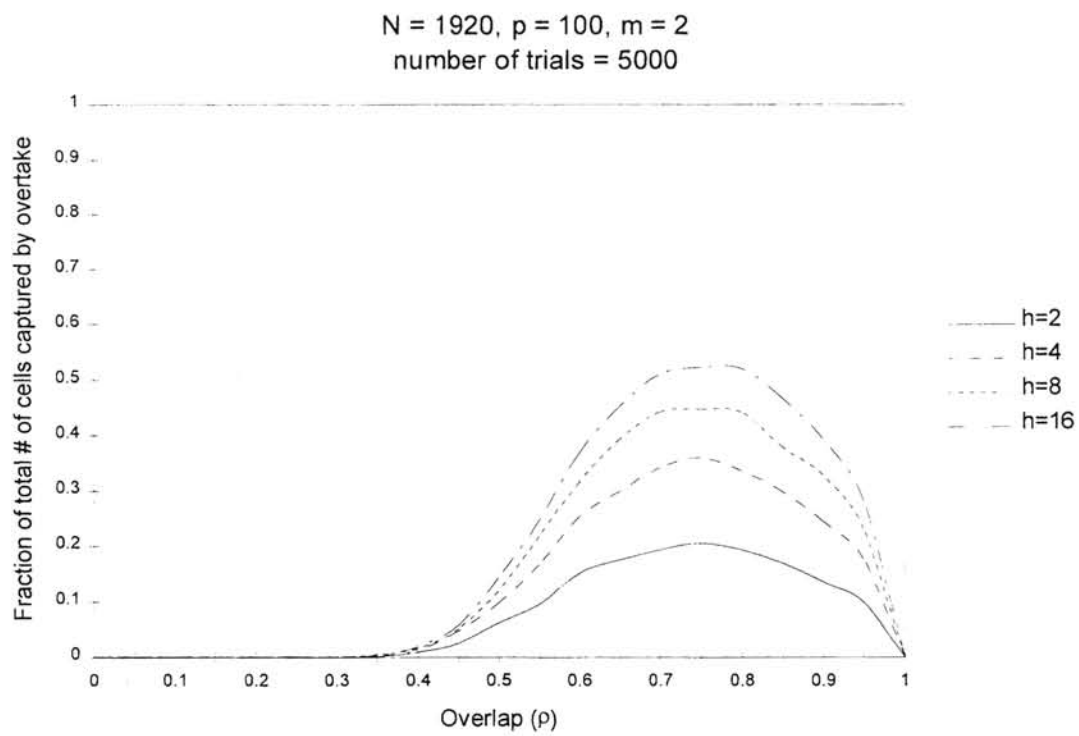


Fig 7.1 Capture by Overtake : $N = 1920, m = 2$

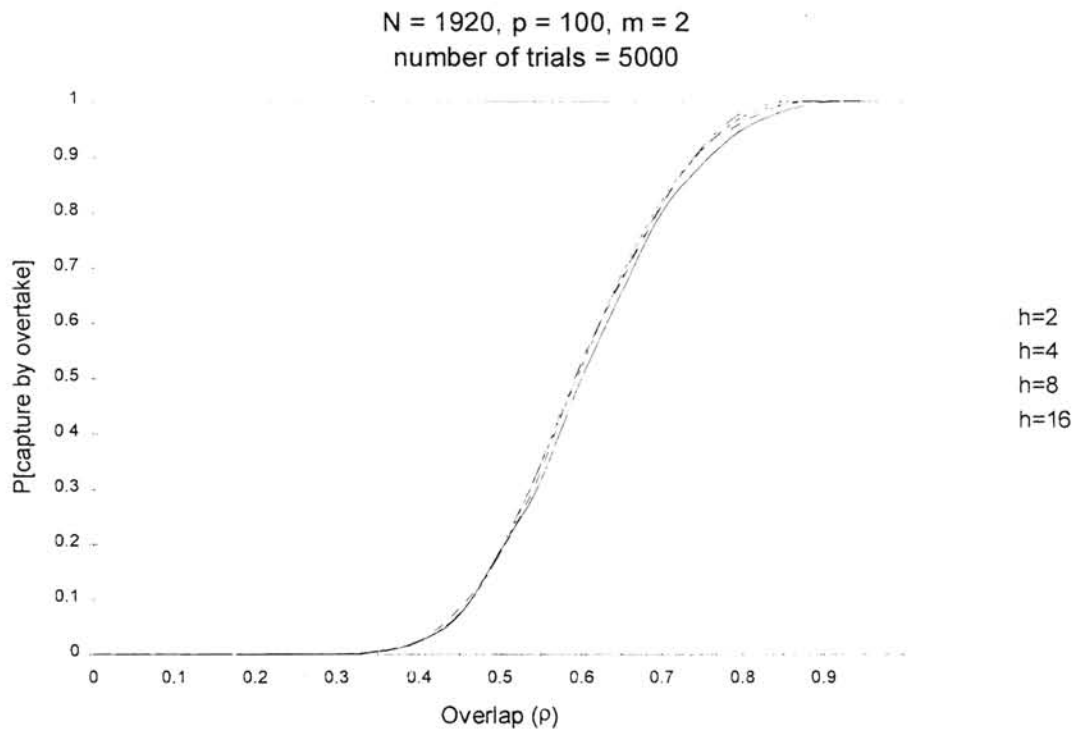


Fig. 7.2 Capture by Overtake : $N = 1920$, $m = 2$

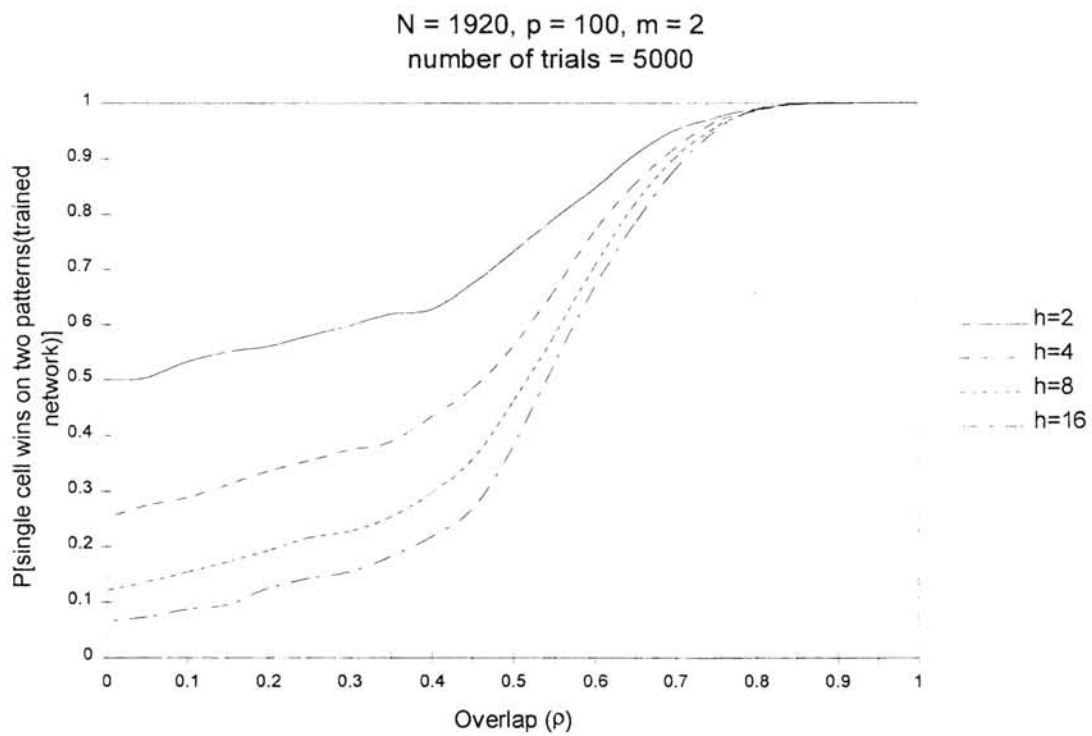
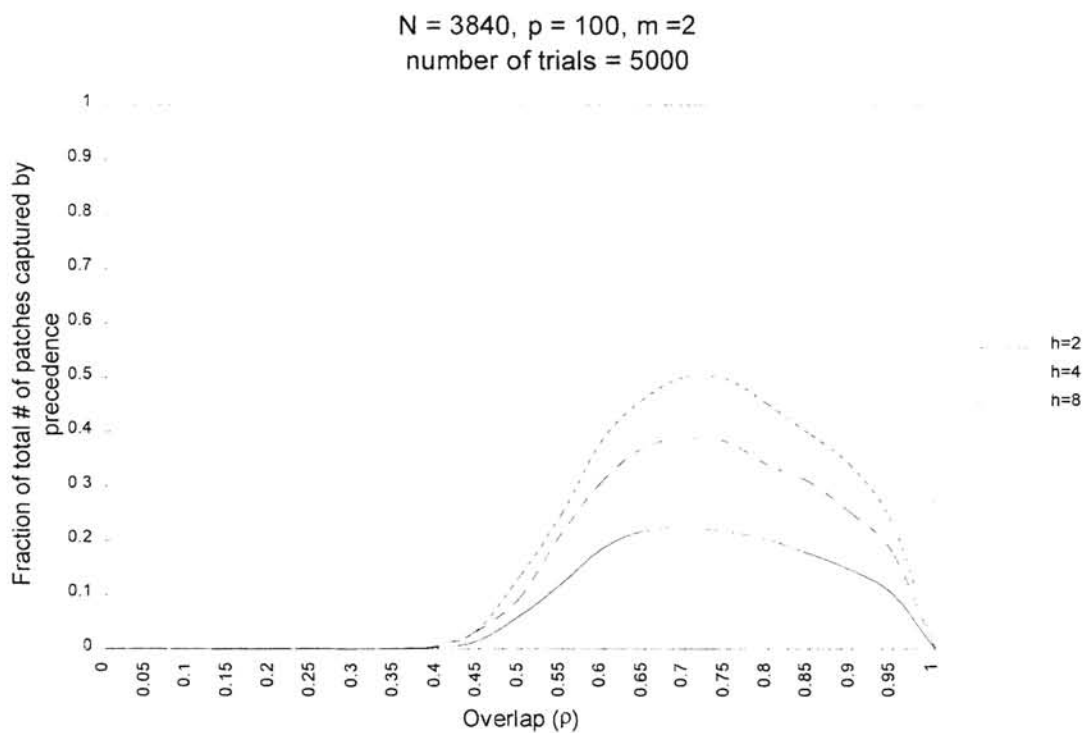
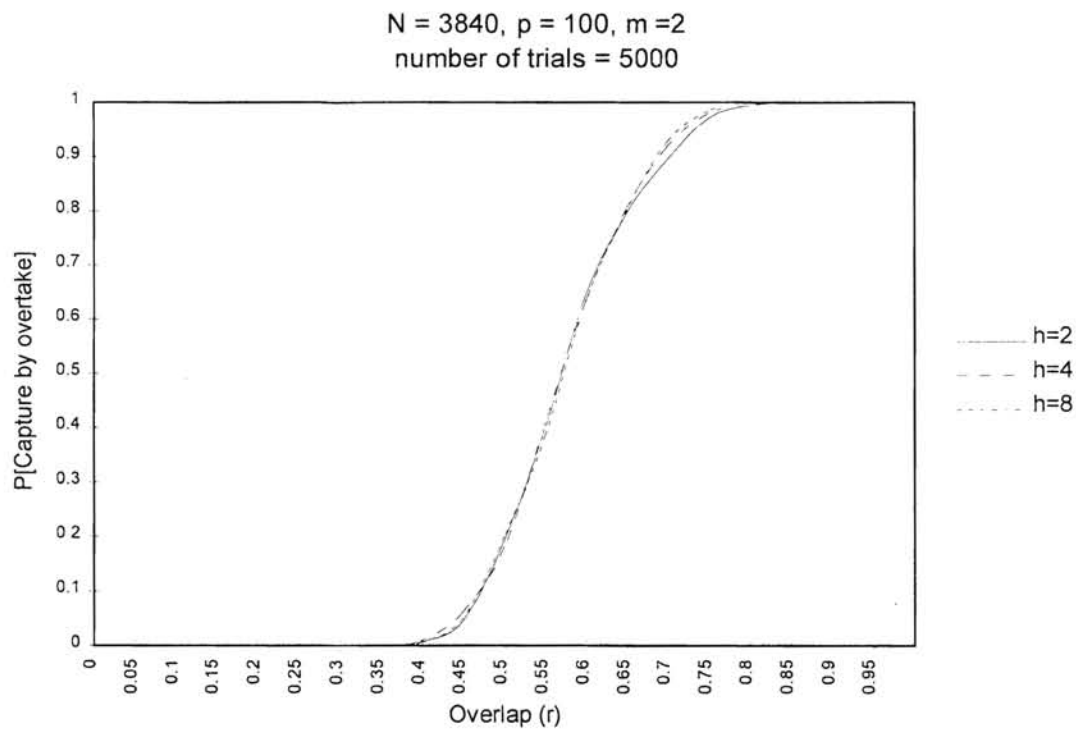


Fig 7.3 Capture by Overtake : $N = 1920$, $m = 2$

Fig 8.1 Capture by Overtake : $N = 3840, m = 2$ Fig. 8.2 Capture by Overtake : $N = 3840, m = 2$

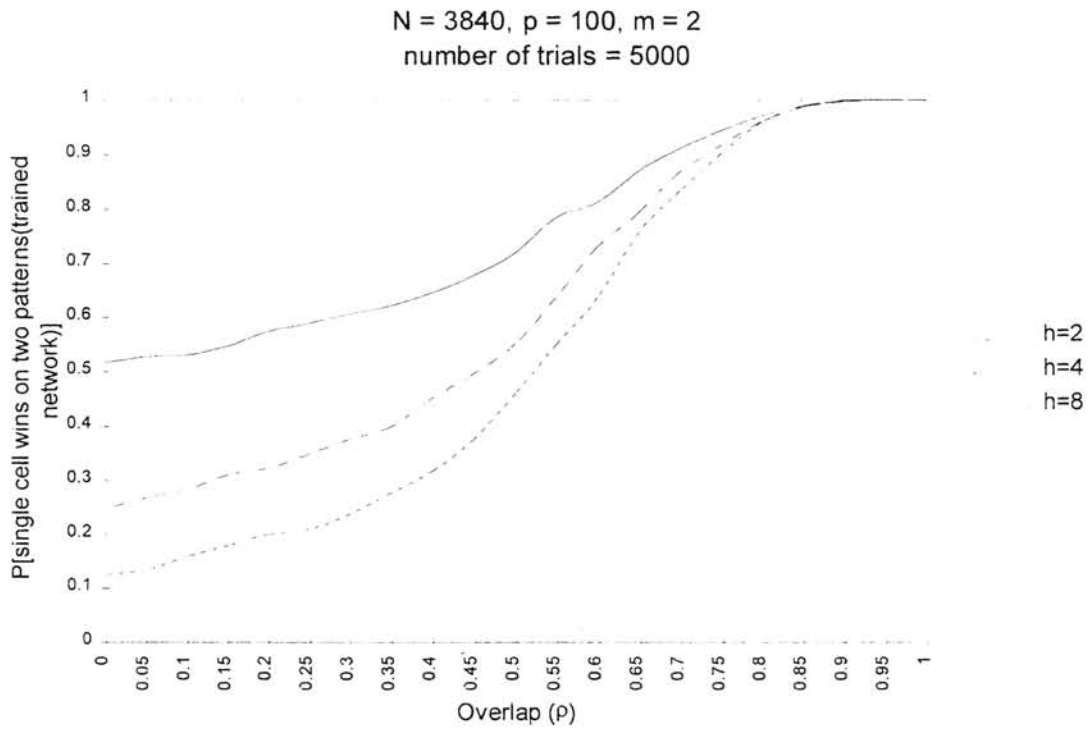


Fig 8.3 Capture by Overtake : N = 3840, m = 2

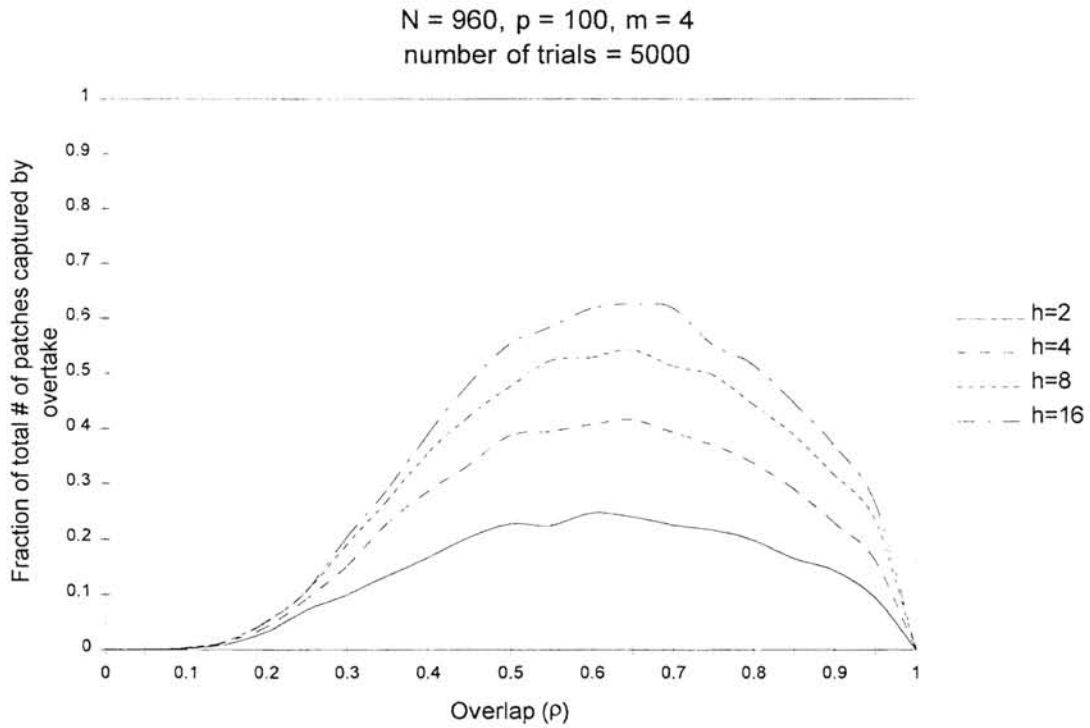


Fig 9.1 Capture by Overtake : N = 960, m = 4

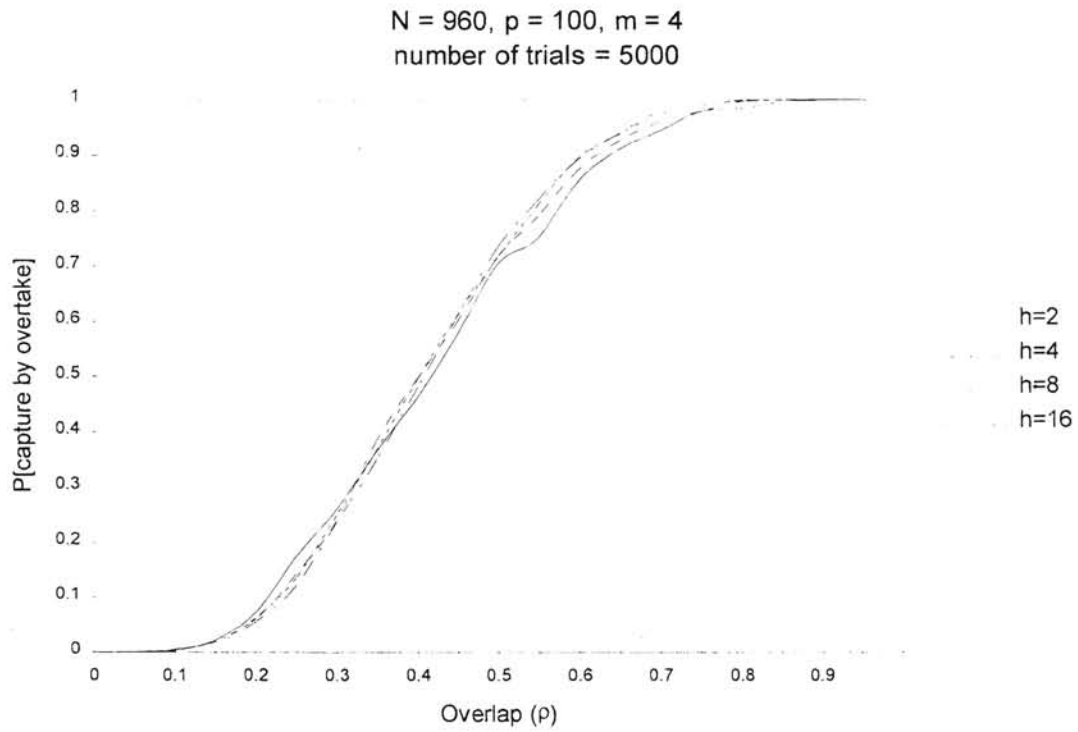


Fig. 9.2 Capture by Overtake : $N = 960, m = 4$

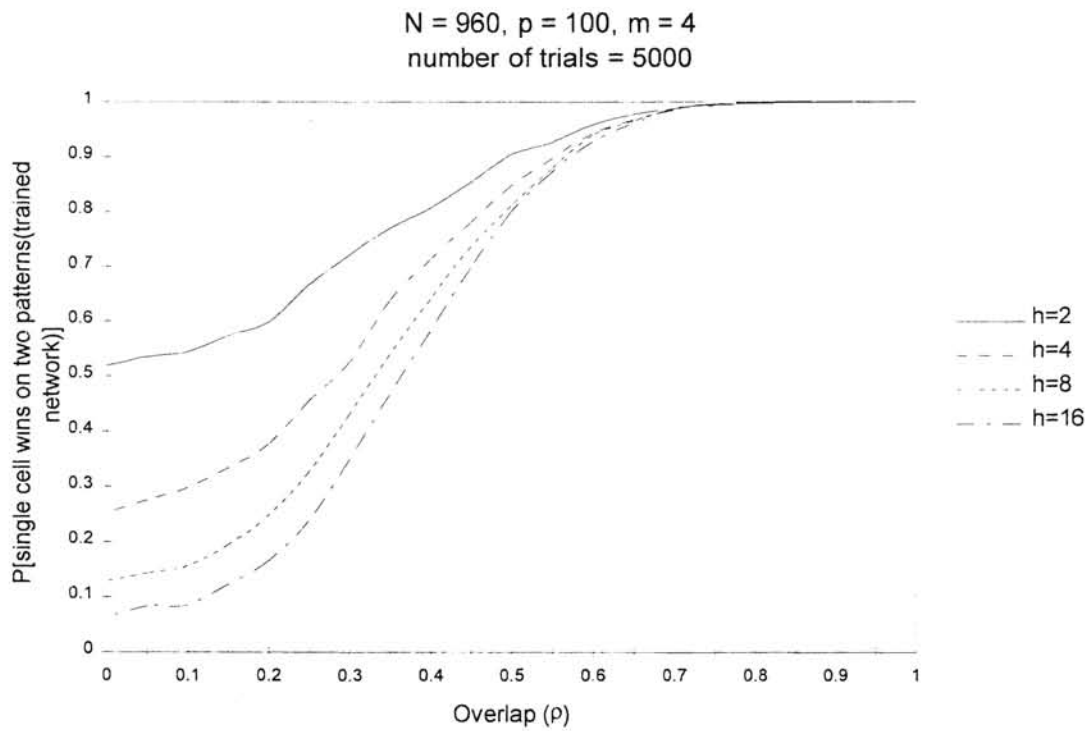
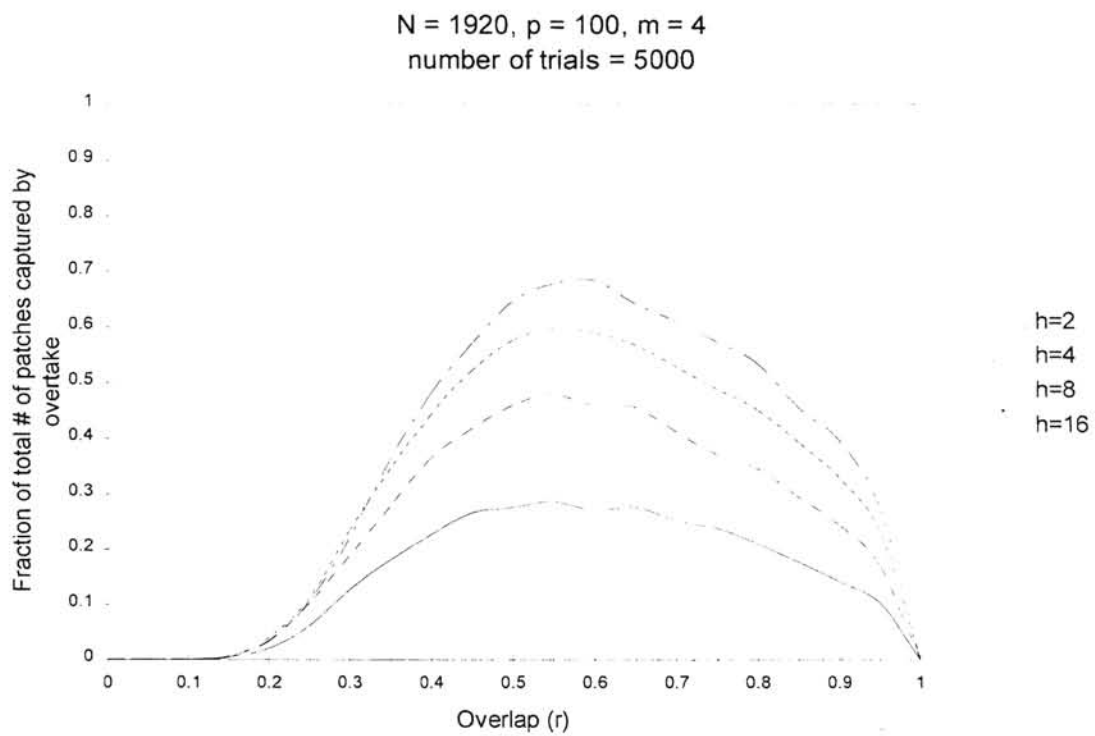
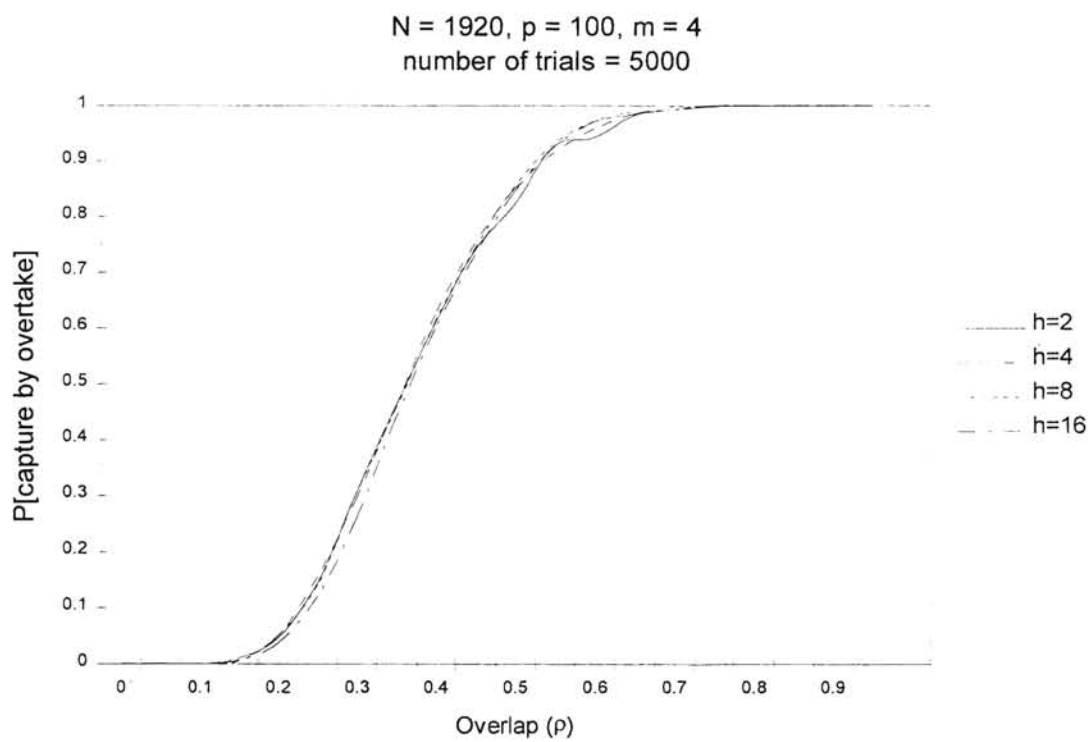
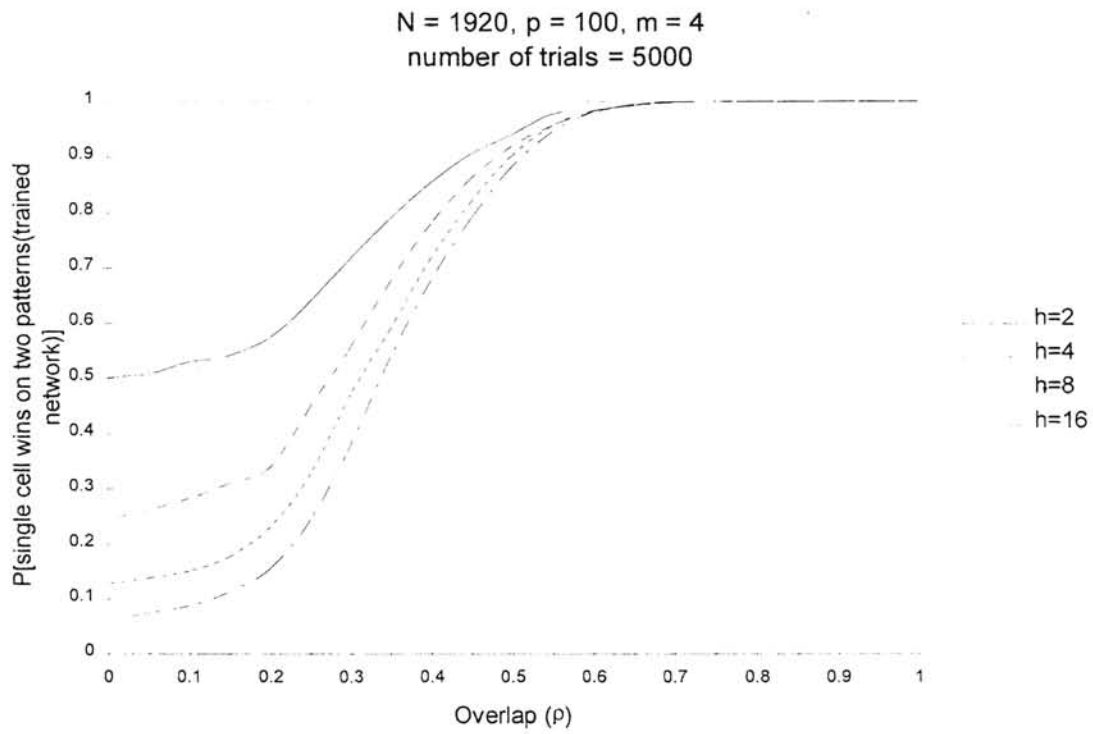
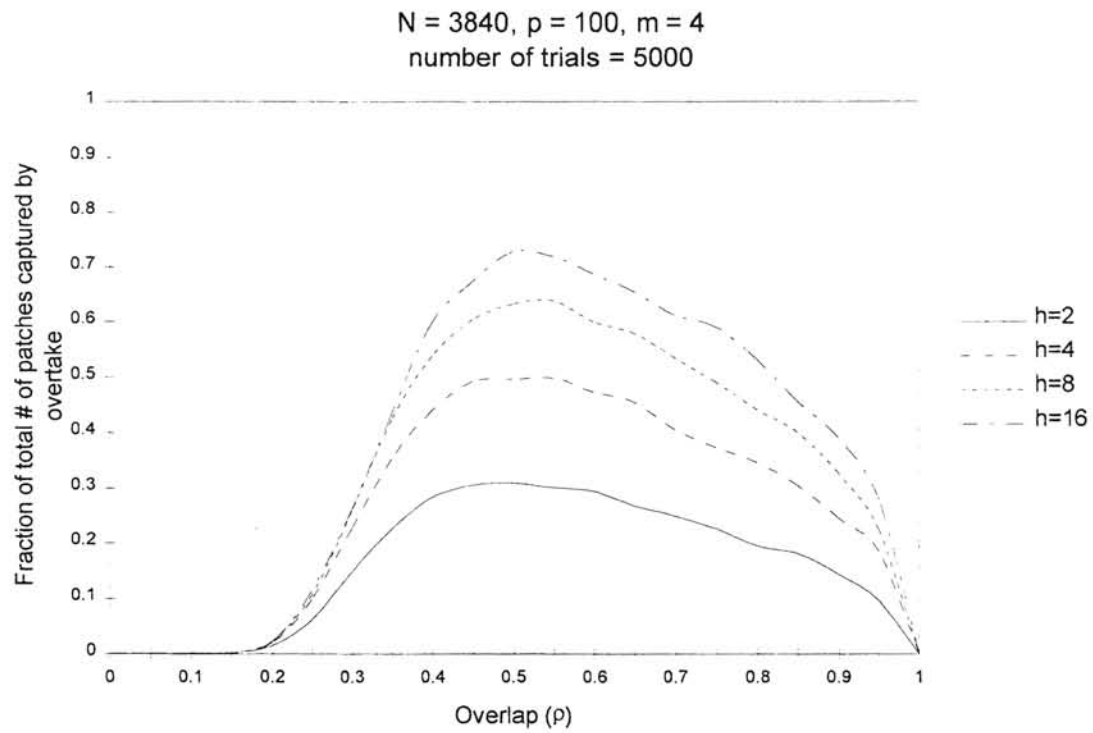


Fig 9.3 Capture by Overtake : $N = 960, m = 4$

Fig 10.1 Capture by Overtake : $N = 1920, m = 4$ Fig. 10.2 Capture by Overtake : $N = 1920, m = 4$

Fig 10.3 Capture by Overtake : $N = 1920, m = 4$ Fig 11.1 Capture by Overtake : $N = 3840, m = 4$

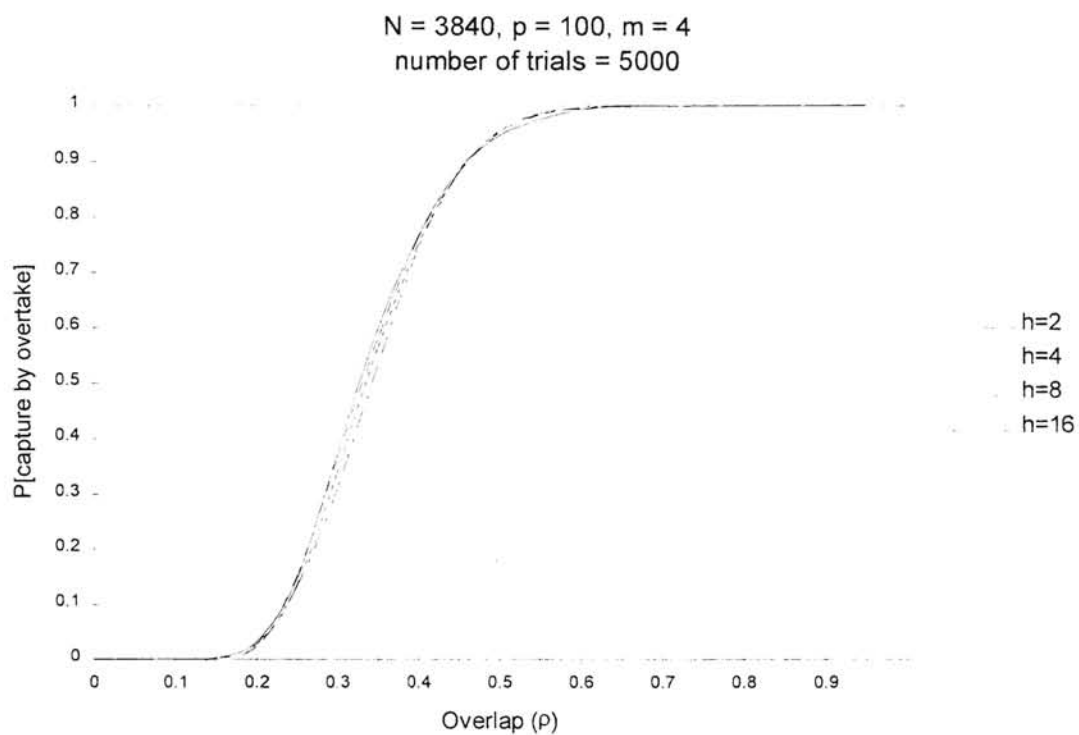


Fig. 11.2 Capture by Overtake : $N = 3840, m = 4$

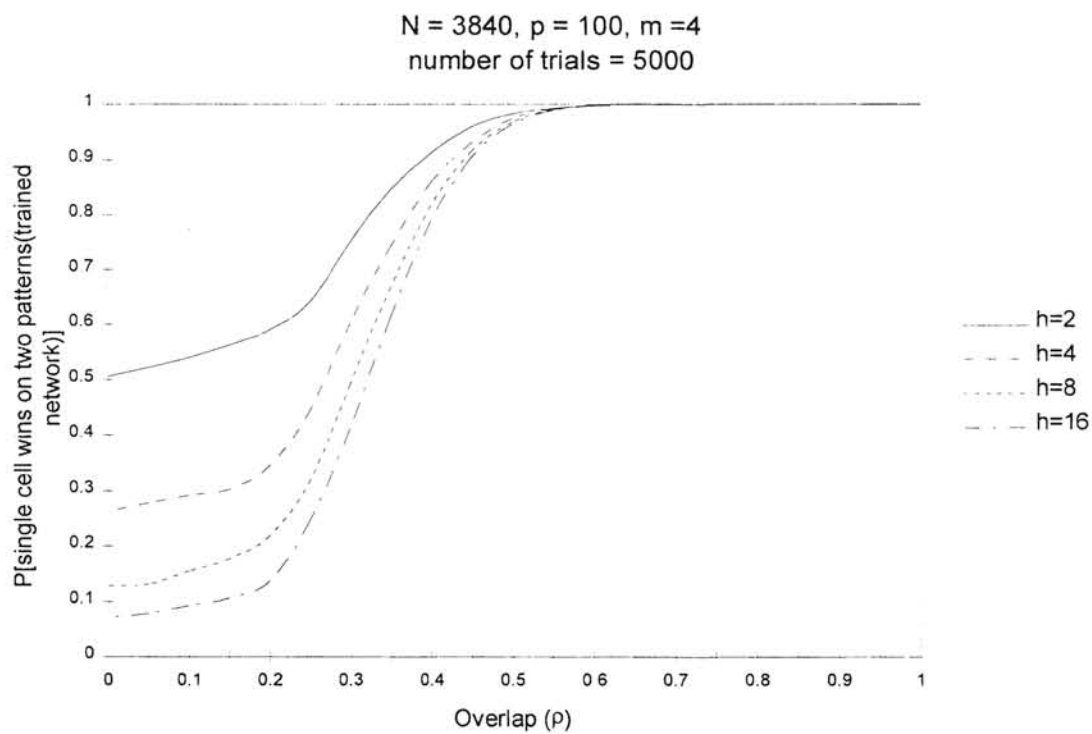


Fig 11.3 Capture by Overtake : $N = 3840, m = 4$

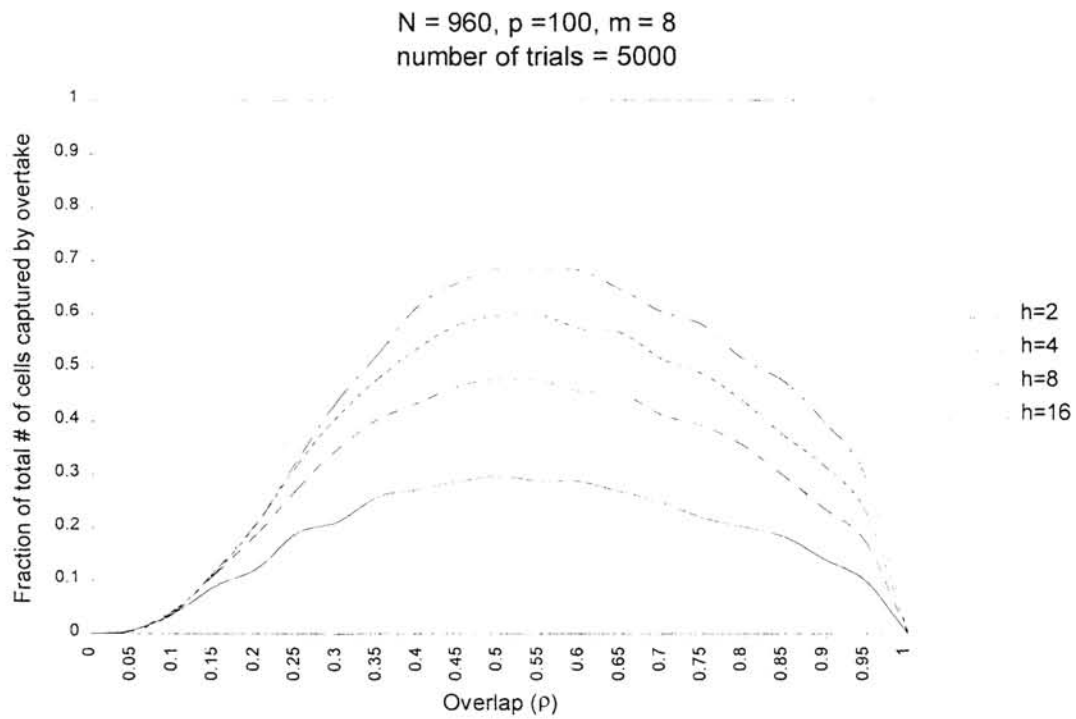


Fig 12.1 Capture by Overtake : N = 960, m = 8

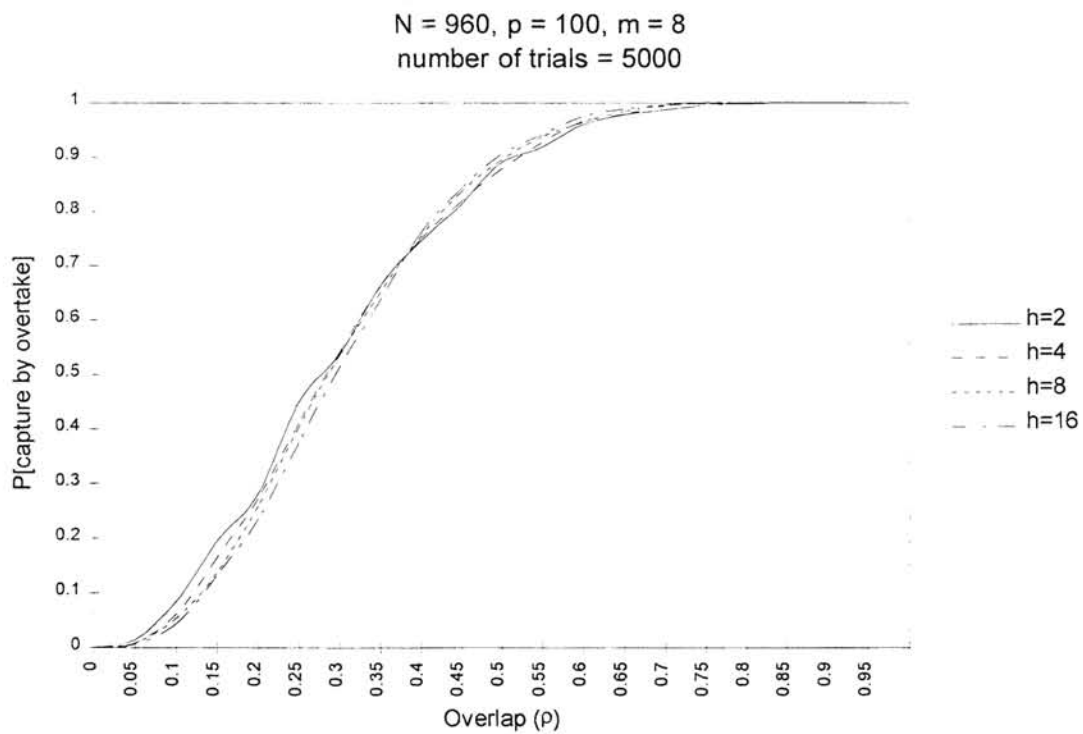
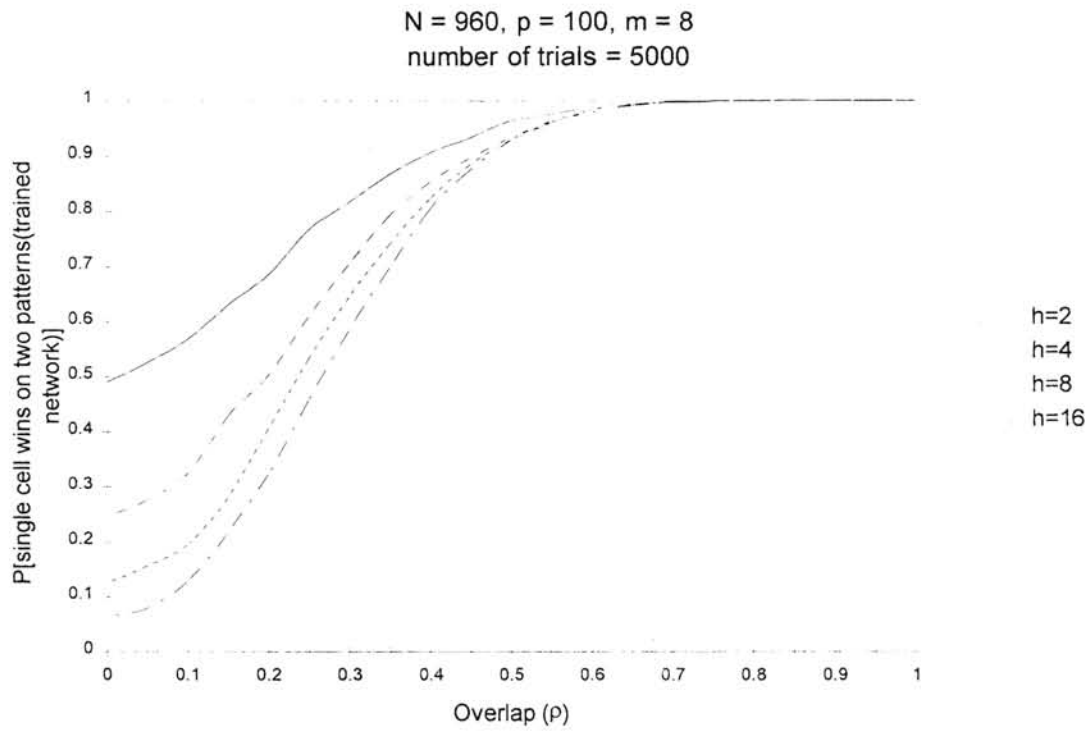
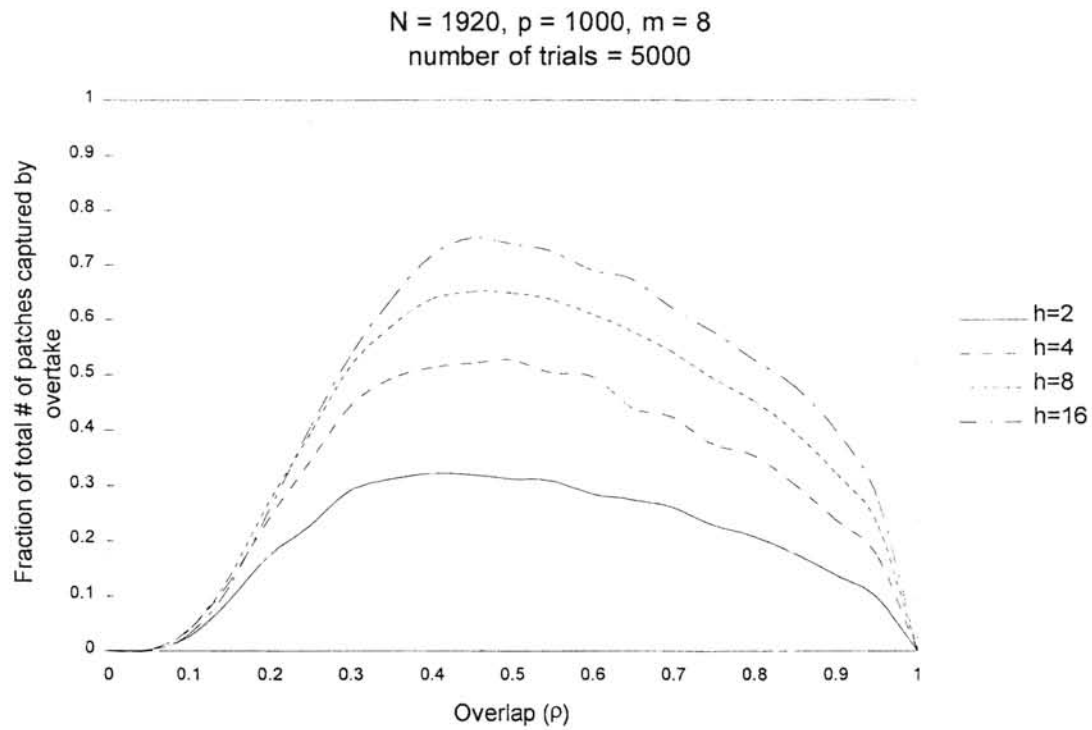


Fig. 12.2 Capture by Overtake : N = 960, m = 8

Fig 12.3 Capture by Overtake : $N = 960, m = 8$ Fig 13.1 Capture by Overtake : $N = 1920, m = 8$

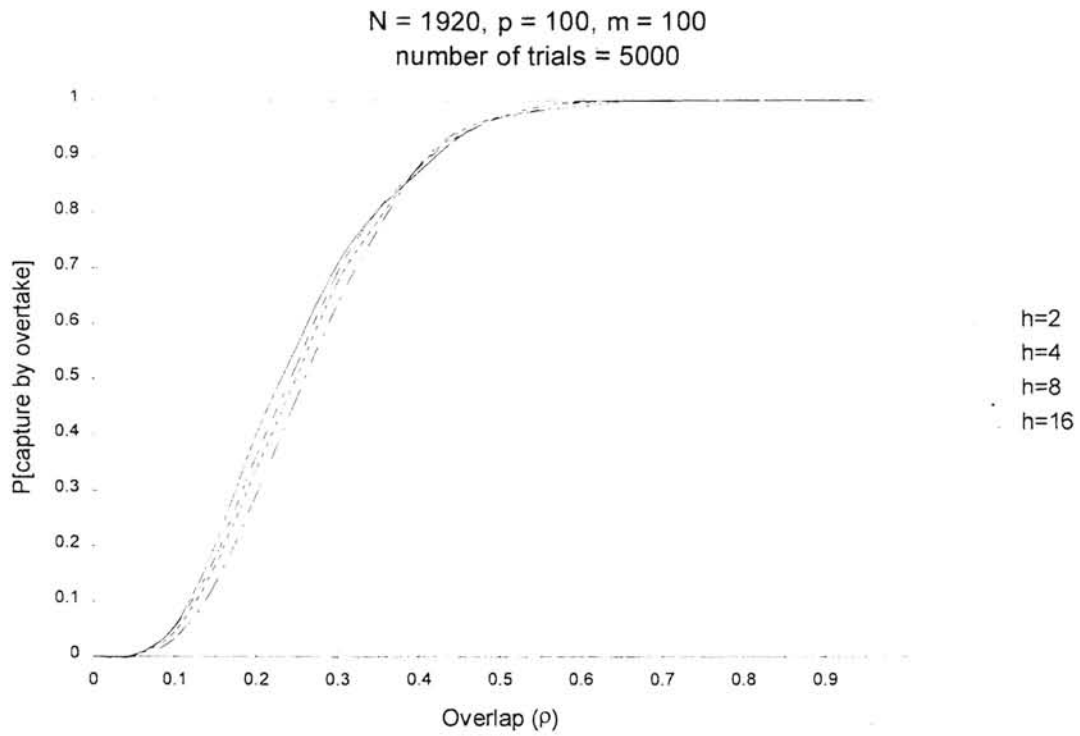


Fig. 13.2 Capture by Overtake : N = 1920, m = 8

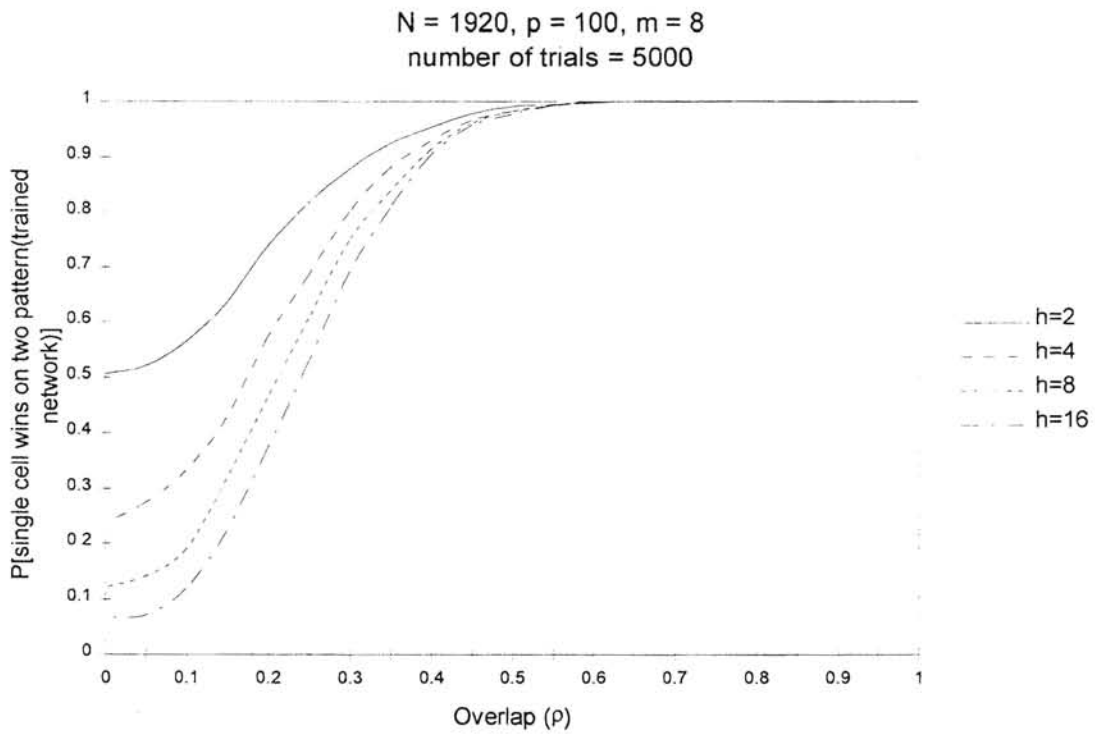
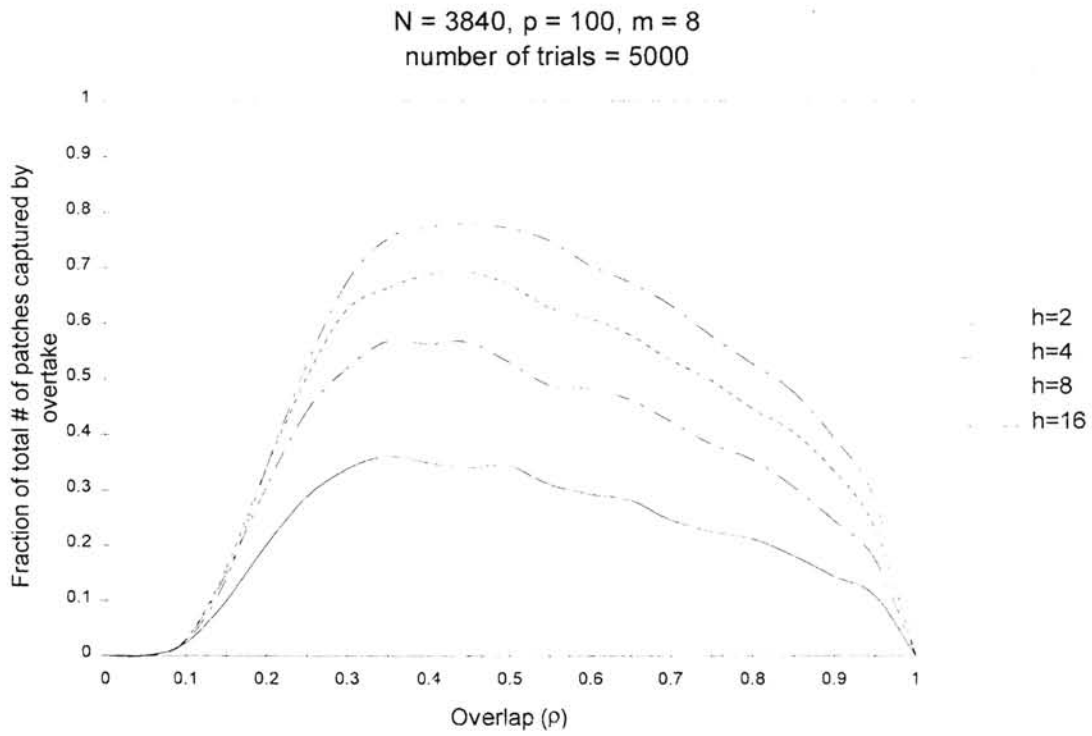
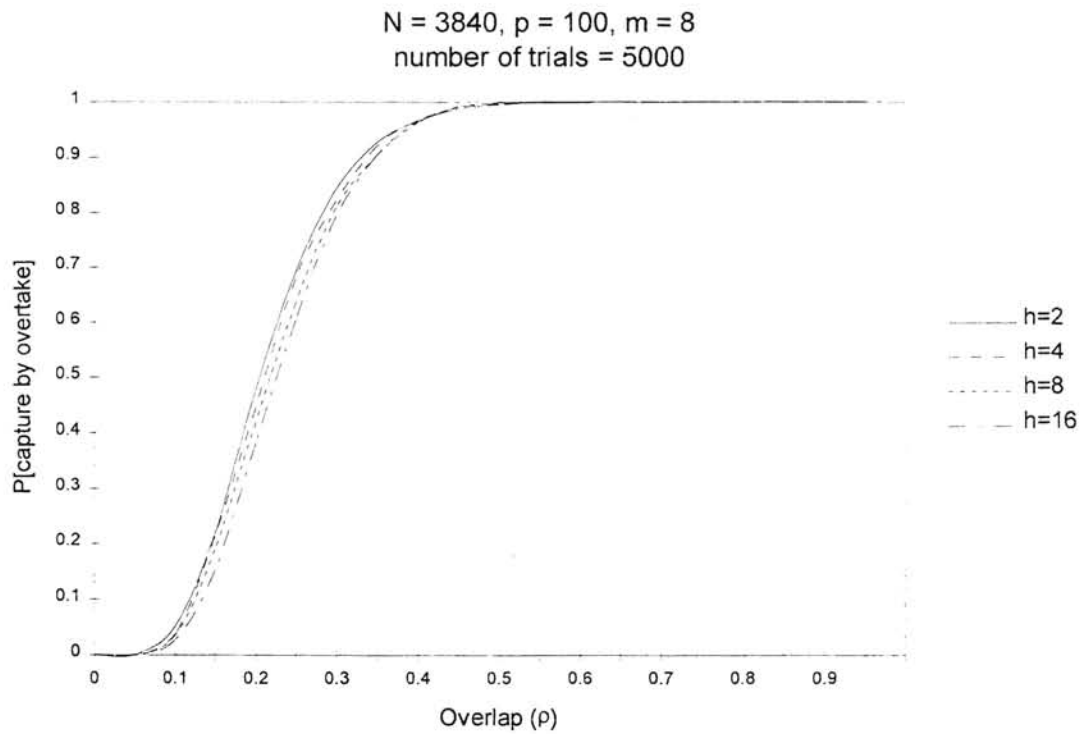


Fig 13.3 Capture by Overtake : N = 1920, m = 8

Fig 14.1 Capture by Overtake : $N = 3840, m = 8$ Fig. 14.2 Capture by Overtake : $N = 3840, m = 8$

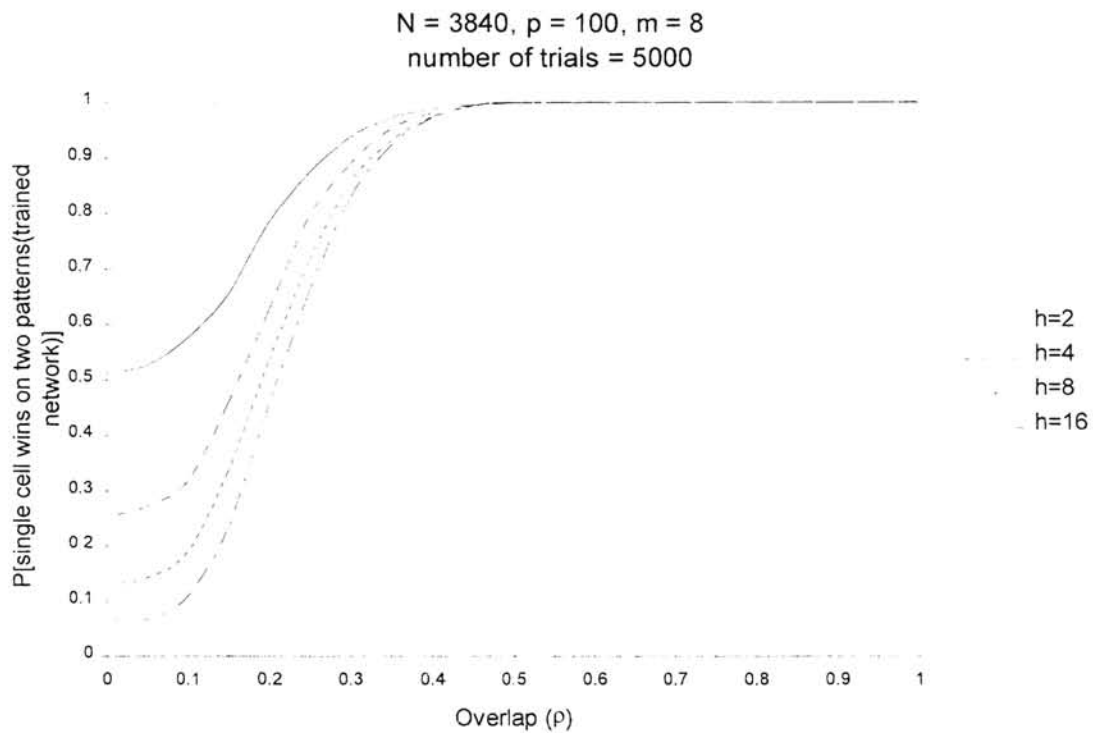


Fig 14.3 Capture by Overtake : $N = 3840$, $m = 8$

To summarize:

- as m increases, more and more cells are captured for all values of overlap ρ between P_1 and P_k .
 - for the same value of m ,
 - as N increases, capture by overtake reduces for $\rho < 1/m$
 - as N increases, capture by overtake increases for $\rho > 1/m$
 - $\rho = 1/m$ (approximately) is the point of intersection of these curves
- In other words, the curves of probability of capture by overtake become steeper as N increases.
- as N increases, the patch size h has less effect on capture by overtake, for very large N , capture by overtake is almost independent of N .

2.5 Review of the Coultrip-Granger Model

Robert L. Coultrip and Richard H. Granger [10] have suggested a model for sparse random networks with LTP learning rules. The main feature of this model is that it implements **linear separability** by statistical means.

2.5.1 Network Description

The basic network is shown in figure 15. The model consists of a single layer of cells that is referred as the *principal layer*. The excitatory cells in the principal layer (principal neurons) receive sparse random innervation by two distinct afferent pathways, a *fixed pathway* and a *plastic pathway*. Axons on the principal neurons form the outgoing *response pathway*. Input pathways are assumed to have some fixed percentage of active lines when the input is present. The inputs to the fixed and plastic pathway may not be of the same dimensions. Although two sets of WTA patches are shown in the block diagram, there is only one set of WTA patches. During *training mode* (to be discussed shortly) it is used by fixed pathway whereas during performance mode it is used by plastic pathway.

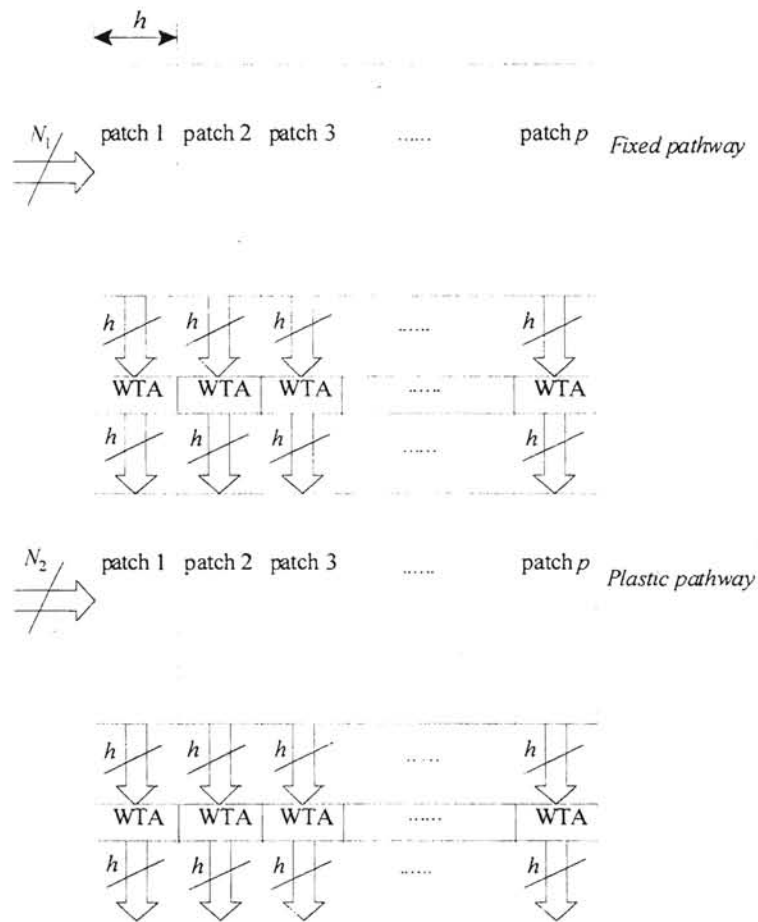


Fig 15 Coultrip-Granger model : Block diagram

2.5.2 Working of the network

The working of the network can be divided into three modes:

1. Preset mode;
2. Training mode;
3. Performance mode.

1. Preset Mode

In the preset mode, the input is impressed on only the fixed pathway. WTA competition selects one cell per patch to fire. The number of distinct input stimuli f_j is assumed to be some finite number N_f . These stimuli on the fixed pathway produce responses r_j .

Since the weights in the fixed pathway are never altered (hence the name fixed pathway), whenever f_j is presented to the it, it will always produce the response r_j . In other words, for every f_j there is a unique r_j which represents the same information as f_j but with different dimension.

2. Training Mode

The general training algorithm for supervised learning of the Coultrip-Granger model is described as following:

The network is presented with (\mathbf{x}, \mathbf{y}) pairs where \mathbf{x} is training datum and \mathbf{y} is its class. The network learns the association. The training data (\mathbf{x}) is presented to the plastic path. The class information (\mathbf{y}) is presented to the fixed pathway. So we have to train with fixed path present, to direct the net what class the datum is supposed to be trained into.

Let there be N_f possible stimuli representing classes. These stimuli are the class information to be presented to the fixed pathway. One of these N_f stimuli is presented to the N_1 lines constituting the fixed pathway and WTA action selects one neuron from each patch. Then the stimulus p_j (training pattern) is presented to the N_2 lines constituting the plastic pathway, and the synapses between active plastic pathway fibers and winning

principal neurons are strengthened via *long term potentiation*, LTP. That is, f_j alone decides which cell in each WTA patch can learn and p_j alone decides which synapses on the winning cell are potentiated. This is desirable because this guarantees that the changing plastic path synapses do not alter the winners selected for some fixed path input f_j over time. Since input f_j always produces output r_j , we can think of this as training the net on an associative input-output pair (p_j, r_j) .

3. Performance Mode

In the performance mode, an input is presented to the plastic path and nothing is presented to the fixed path. The output of the WTA patches will indicate the code of the class to which this input belongs.

2.5.3 Advantages and disadvantages of the network

Since the fixed pathway determines the cells to be trained, there is no undesired capture by precedence. But still there will be some undesired capture by overtake in the performance mode if the training data set is not fair.

Also, since class information is presented to the fixed pathway, *a priori* knowledge of the classes/clusters is necessary.

2.5.4 Similarity and difference with Kohonen network

In a Kohonen network, the weight vector becomes more and more similar to the input vector as the learning progresses. In other words, component of the weight vector along the input vector increases as the learning progresses thereby reducing the size of the error or difference vector. In Coultrip-Granger model, the component of the weight vector along the input vector increases as the network is trained but the size of the error/difference vector remains same. Still the angle between the input vector and the weight vector reduces.

Like the Kohonen network, the class information is needed before training can be initiated. However the learning rule (LTP) is totally different from the Kohonen rule.

2.6 Proposed Model

In the proposed model, an attempt has been made to overcome the disadvantages of the Coultrip-Granger model.

For our purpose, we have proposed two changes for an electronic implementation of a modified Coultrip-Granger model. The modifications in the model are as follows:

1. The fixed and plastic pathways are assumed to be exactly identical. The sparsity of these weight matrices is assumed to be q .
2. The same input is presented to both the pathways. A fraction r of the N input lines (i.e. Nr lines) are active at any time the input is present.

The network is shown in Fig. 16. In this network, unlike the Coultrip-Granger model, there are **two** sets of WTA patches.

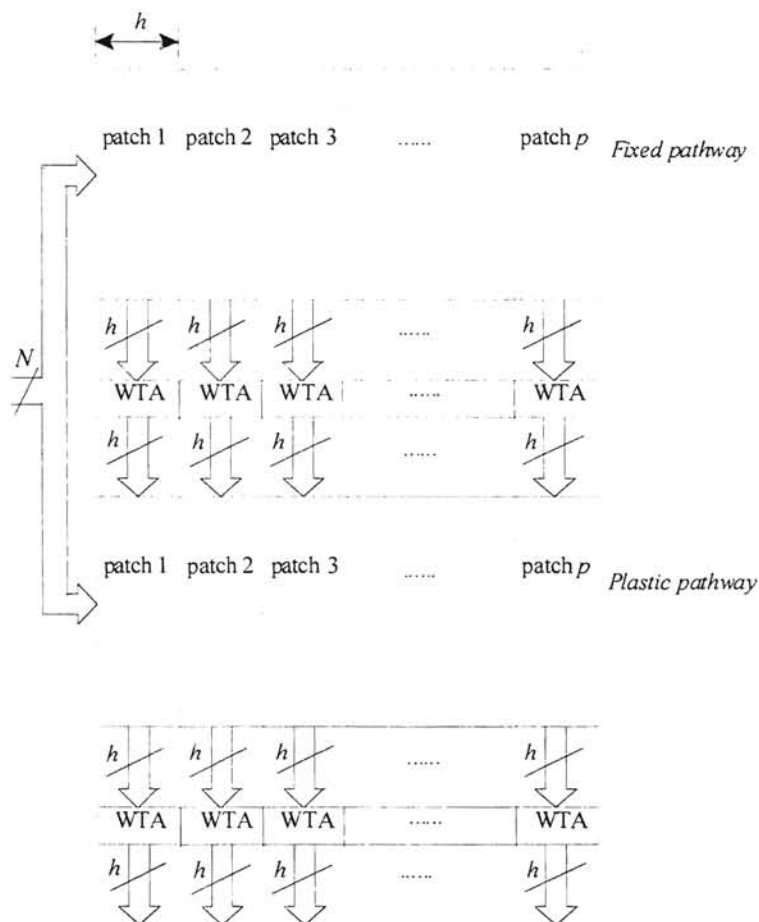


Fig. 16 Proposed model : Block diagram

2.7 Proposed Training Algorithm

The input signal is presented to both, fixed and plastic, pathways. The winners of both the pathways are determined. The outputs of the winning cells are set to one and other outputs are set to zero. The output binary strings of both the pathways are XORed

and sum of the bits in the resultant string is computed. Let 'z' be the sum of the bits in the resultant string.

We define a cluster as a set of vectors having a hamming distance less than or equal to some predefined distance from the center of the cluster. In our network, the vector presented first becomes the center of the cluster. Let H be the allowable hamming distance from the center for any vector to belong to the cluster, i.e., the maximum number of mismatching 1's between the center of the cluster and any other vector in the same cluster is $H/2$. This condition sets the minimum required overlap of active lines between the center of the cluster and any other vector in the cluster. Let the minimum required overlap be ' ρ_{\min} '.

$$\rho_{\min} = \frac{Nr - \frac{H}{2}}{Nr} = 1 - \frac{H}{2Nr} \quad (2.1)$$

As we have seen earlier (section 2.3) , for the naive weight matrix, there exists a finite probability, P_n , that a single cell wins on two different patterns having an overlap ρ_{\min} between them. P_n is a function of the patch size h as well. The larger the patch size, the lower is P_n . From this information we can determine the overlap between the outputs produced by the two vectors when presented to the naive weight matrix.

$$\begin{aligned}
\text{output overlap} &= \frac{\text{number of matching 1's between output strings}}{\text{number of patches}} \\
&= \frac{\text{number of patches} \times P_n}{\text{number of patches}} \\
&= P_n
\end{aligned}$$

The vectors producing output codes having an overlap greater than or equal to P_n with the output code produced by the center of the cluster, by definition, belong to that cluster.

Now consider the plastic pathway. Let's assume that the network is already trained on one of the input vectors. Now a new vector is presented to the network. There are two possibilities: either this vector belongs to the cluster defined by the first training vector or it does not. To determine this, we will have to set some criterion based on the outputs of the two weight matrices (fixed and plastic). We know that there exists a probability that a single cell wins on two patterns after the network is trained on the first pattern. Let this probability be P_l for the overlap between the two input patterns equal to ρ_{\min} which is defined above. This probability is greater than P_n (defined above) because of the fact that some of the cells are captured by "capture by precedence." As we have seen, The probability of capture by precedence is a function of the overlap between the two patterns. Let $P[\text{capture by precedence}] = P_{cp}$ for $\rho = \rho_{\min}$. So the number of captured cells will be $p \times P_{cp}(1 - P_n)$.

The network does not know the overlap between the two input vectors, but it can calculate the number of captured cells. If the output strings of the two WTA patches are XORed, the number of non-zero bits, z , is twice the number of captured cells. Which gives

$$\text{number of captured cells} = z/2$$

This number is a function of the overlap ρ between the two input vectors, the present input and the input on which the network was trained. If $z/2 \geq p \times P_{cp}(1-P_n)$, it indicates that $\rho < \rho_{\min}$ and the vector belongs to the cluster and training is needed. If $z/2 < p \times P_{cp}(1-P_n)$, it indicates that ρ may or may not be greater than ρ_{\min} . (Refer to figures 3.1, 4.1 and 5.1). This indicates that the network may or may not have been trained on the same or similar pattern.

To determine, if the network has already been trained on the present pattern, the activations of the winning cells producing matching 1's should be checked as discussed below. The winning cells producing matching 1's are termed as "common winners".

Out of p winners, let z' winners be common. For the fixed pathway, let the activations of the p common winners be $x_{f1}, x_{f2}, \dots, x_{fz}$. For the plastic pathway, let the activations of the p common winners be $x_{p1}, x_{p2}, \dots, x_{pz}$.

The differences between the activations x_{pi} and x_{fi} are calculated. If the network has already been trained on the present input or on the center of the cluster it belongs to, all the coactive synapses must be trained at least once, i.e., if the network has been trained on the same input,

$$x_{pi} - x_{fi} \geq \Delta w \cdot x_{fi}, \quad i=1, 2, \dots, z', \quad (2.2)$$

or if the network has been trained on the center of the cluster the present input vector belongs to,

$$x_{pi} - x_{fi} \geq \rho_{\min} \cdot \Delta w \cdot x_{fi}, \quad i=1, 2, \dots, z'. \quad (2.3)$$

If the above condition is satisfied, training is not needed. Otherwise training is needed.

On the other hand, if the sum of the bits in the ANDED string is less than P_1m , obviously, training is needed.

The training algorithm discussed above is also shown in the flow-charts (fig. 17, 18 and 19). Fig. 17 shows the general training algorithm. Fig. 18 is the expanded version of the flow-chart in fig 17. Fig. 19 is the expanded version of the function TRAIN in fig. 18.

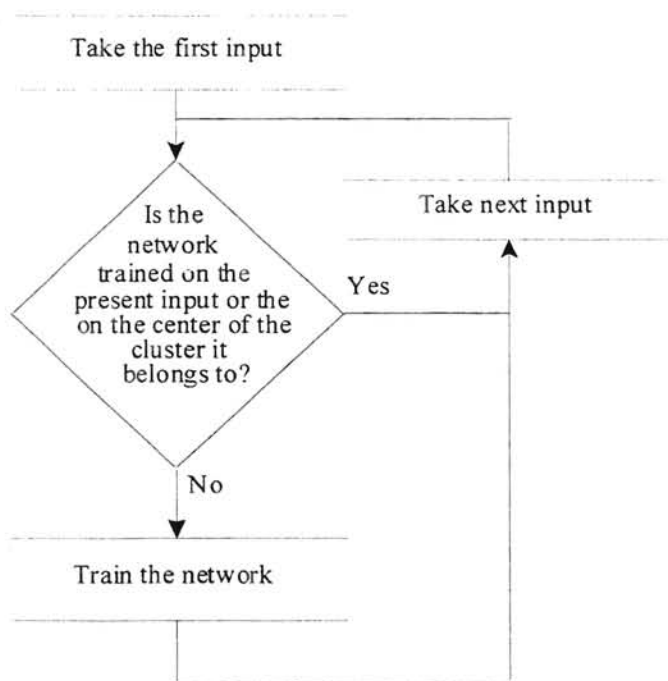


Fig. 17 Training algorithm : Simplified flow chart

TRAINING ALGORITHM

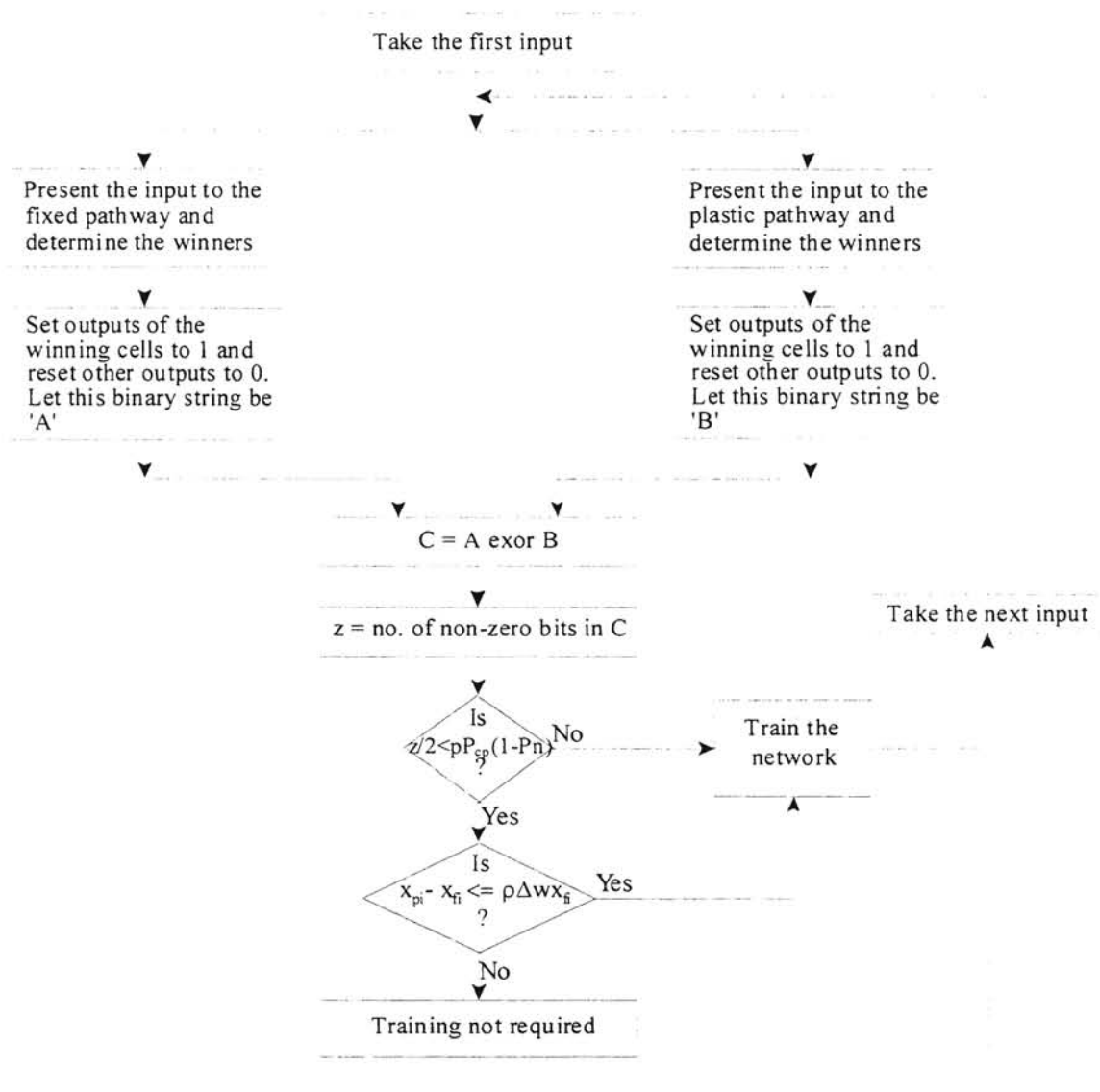


Fig. 18 Training algorithm : Flow chart

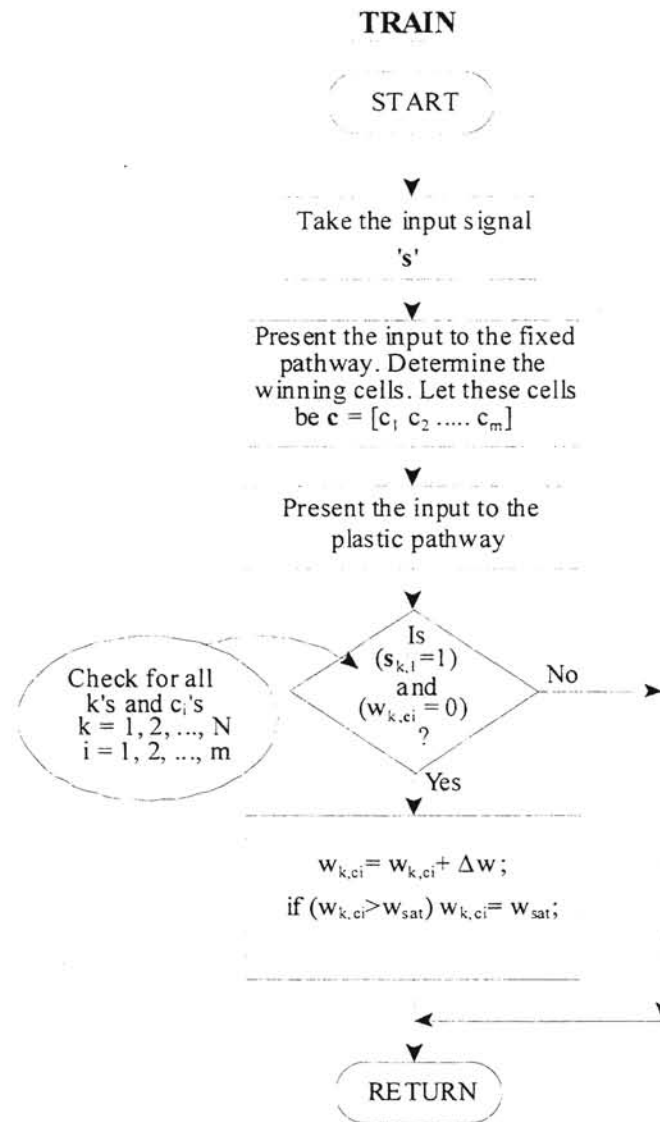


Fig 19 Training algorithm : TRAIN routine

2.8 Advantages of the proposed model over the old model

1. Capture by Precedence

In the proposed model, the cells to be trained are always determined by the winners in the fixed pathway (naive weight matrix), there will not be any **undesired** capture by precedence.

2. Capture by Overtake

The proposed model is not affected by capture by overtake since before training the network, we are checking if the network was trained on similar pattern. If it has been trained, we are not training it again by rule.

3. *A priori* knowledge of the classes is not needed

Since we are presenting the same data to both the pathways, *a priori* knowledge of the classes/clusters is not needed.

3. Control over cluster dimension

Since the cluster dimension is user defined, we have a control over the quantum for sampling the hypersurface.

4. Extensibility to LVQ Network

An additional layer of network, can extend this model to LVQ network, thereby enabling to form concave or linearly inseparable regions on the hypersurface. This is not true for the Coultrip-Granger model since that model is not free from capture by overtake. In performance mode it can cause errors.

2.9 Simulations

2.9.1 Experiment 1

In this experiment, it is assumed that we know the ideal data set. In other words, we know the cluster centers. N , p , h , ρ_{\min} are set to 960, 30, 16 and 0.9 respectively and following procedure is followed:

1. The network was trained on 10 different randomly generated non-orthogonal patterns.
2. Test data consists of (i) 195,000 randomly generated uniformly distributed patterns and (ii) 500 patterns from each cluster. The picture will look somewhat like figure 20.

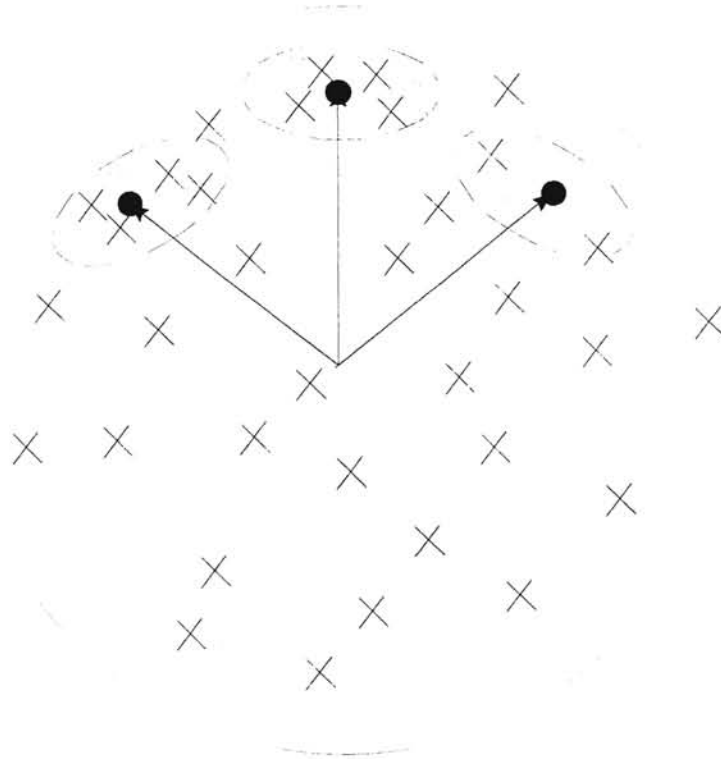


Fig. 20 Experiment 1

For simplicity, only three clusters are shown. The solid circles show the cluster centers (training data set). The \times 's show the randomly generated and uniformly distributed data set and the hollow circles show the vectors from that cluster.

3. The error is calculated. There can be two types of error: (i) the vector not belonging to the cluster is classified as belonging to the cluster; and (ii) the vector belonging to the cluster is classified as not belonging to the cluster.

Only the second type of error is found. There is 0.14% error rate.

2.9.1 Experiment 2

In this experiment, unlike the experiment 1, ideal training set is not used. N , p , h , ρ_{\min} are set to 960, 30, 16 and 0.8 respectively and following procedure is followed:

1. The network was trained on four training patterns from each of the ten clusters. But for training ρ_{\min} is calculated as follows: $\rho_{\min\text{-train}} = \rho_{\min} + (1 - \rho_{\min})/2$. In other words, the training data set is chosen from a tighter cluster within the cluster defined by ρ_{\min} .
2. Test data consists of (i) 195,000 randomly generated uniformly distributed patterns and (ii) 500 patterns from each cluster. The picture will look somewhat like figure 21.

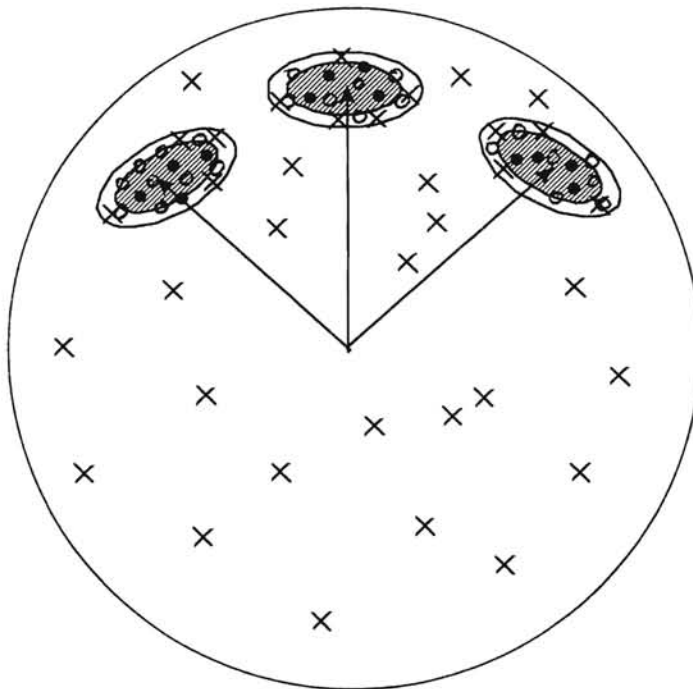


Fig. 21 Experiment 2

Again for simplicity, only three clusters are shown. The solid circles show the cluster centers (training data set). They belong to the shaded region within the cluster, i.e. they have a tighter distribution within the cluster. The 'x's show the randomly

generated and uniformly distributed data set and the hollow circles show the vectors from that cluster.

3. The error is calculated. There can be two types of error: (i) the vector not belonging to the cluster is classified as belonging to the cluster; and (ii) the vector belonging to the cluster is classified as not belonging to the cluster.

Only type (i) error is found. Patterns in the vicinity ($\rho=0.78$) are identified as belonging to the cluster resulting in 0.0875% error rate.

2.9.3 Conclusion

From the simulation results, we conclude that the network clusters the input data. Since the cells to be trained are determined by the fixed pathway, there is no undesired capture by precedence. Also by tightening the domain of training data set and the region of influence of each training vector, we ensure that there is no undesired capture by overtake. Since the same information is presented to both the pathways, the class information is not required. The user is required to define the cluster size thus has a control over it.

2.10 Reconstructing the template

R. Meyyappan [10] has discussed the concept of reconstructing the template by backpropagation through the trained weight matrix. But as the network is trained on more number of patterns, it becomes difficult to determine the threshold.

Since, in the proposed model, we have two weight matrices available, an attempt is made to reconstruct the template by backpropagation through fixed weight matrix. As there are no weight changes in the fixed weight matrix, it seems to be easier to determine the threshold.

The expected number of the coactive synapses on the winner cells, μ_w , and the deviation σ_w are a function of patch-size h . They are not readily determined in closed form [10,12]. But from simulations, we can determine μ_w and σ_w for different values of h .

For $N=960$, following values of μ_w and σ_w are obtained:

h	μ_w	σ_w
4	20.55	2.9
8	22.25	2.6
12	23.05	2.5
16	23.75	2.4
20	24.10	2.3

Fig. 22 shows the weight matrix and the LOT vector. For simplicity, all the active lines of the LOT vector are grouped together.

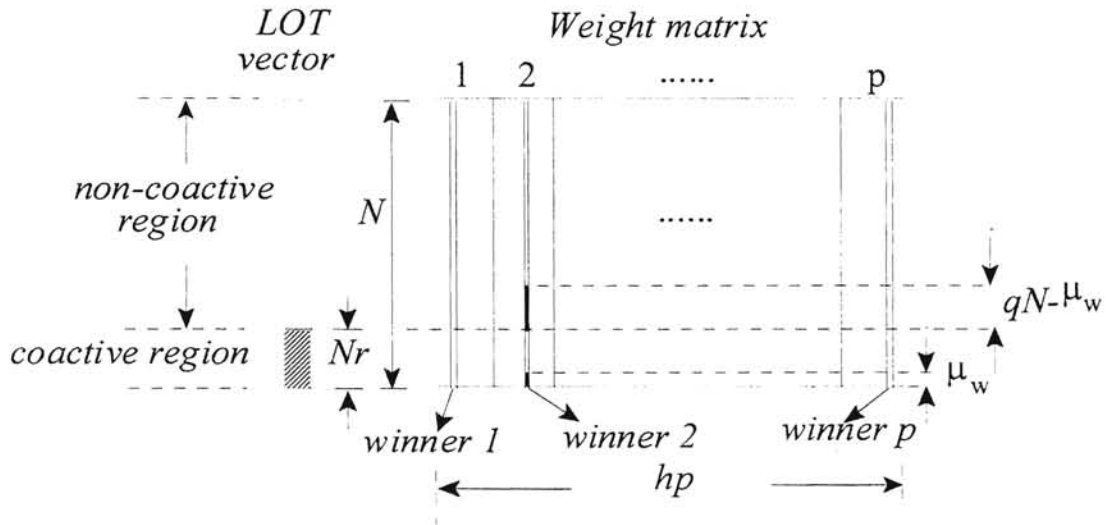


Fig. 22 Backpropagation through fixed pathway

The sparsity of the weight matrix q is 0.1 and it will be maintained on the winner cells also, although it might be high in some region and low in another. Sparsity of the weight matrix (winner cell) in both the coactive and non-coactive regions is defined as follows:

$$\text{Sparsity of the weight matrix (winner cell) in coactive region} = \frac{\mu_w}{Nr} = q_c \quad (2.4)$$

$$\text{Sparsity of the weight matrix (winner cell) in non-coactive region} = \frac{qN - \mu_w}{N(1-r)} = q_n$$

(2.5)

When the signal is backpropagated through the weight matrix, all 'p' winner columns get added. The distribution of the synapses is binomial. Thus the expected value of the sum of the weights in one column of the weight matrix is readily calculated.

In coactive region,

$$E [\text{sum of weights in one row}] = \mu_c = pq_c = p \frac{\mu_w}{Nr} \quad (2.6)$$

$$\sigma_c^2 = pq_c(1 - q_c) \quad (2.7)$$

In non-coactive region,

$$E [\text{sum of weights in one row}] = \mu_n = pq_n = p \frac{qN - \mu_w}{N(1-r)} \quad (2.8)$$

$$\sigma_n^2 = pq_n(1 - q_n) \quad (2.9)$$

Since the network does not know which part is coactive and which region is non-coactive, it is advisable to set some threshold and call the subthreshold sums as belonging to the non-active region and set them to zero and call the others as belonging to the coactive region. In doing so, we are losing some data and adding some noise due to the variances associated with the expected values. To minimize this error, we should choose optimum decision boundary between μ_n and μ_c as the threshold. This decision boundary is given by solving the equation:

$$2 \ln \left(\frac{P_n \sigma_c}{P_c \sigma_n} \right) = \left(\frac{d - \mu_n}{\sigma_n} \right)^2 + \left(\frac{d - \mu_c}{\sigma_c} \right)^2 \quad (2.10)$$

where P_n is the probability of a bit belonging to the non-coactive region and P_c is the probability of a bit belonging to the coactive region.

Let's assume that only 40% of the mitral patches are active at any time. Now in active region P_n is equal to P_c . Now the decision boundary can be given by solving the following equation for d .

$$2 \ln \left(\frac{\sigma_c}{\sigma_n} \right) = \left(\frac{d - \mu_n}{\sigma_n} \right)^2 + \left(\frac{d - \mu_c}{\sigma_c} \right)^2 \quad (2.11)$$

For patch-size $h = 16$, and number of patches $p = 30$, following results are obtained analytically. Results obtained by simulations agree with the theoretical results.

$$\mu_c = 3.71 \quad \sigma_c = 1.80$$

$$\mu_n = 2.82 \quad \sigma_n = 1.59$$

$$d = 3.61$$

NOTE

If μ_n and μ_c are to be separated by $\sigma_c + \sigma_n$ (this is equivalent to 3dB SNR) we can determine the minimum number of patches p required to achieve this by solving the following equation for p

$$\mu_c - \mu_n = \sigma_c + \sigma_n, \quad (2.12)$$

If we substitute values of μ_c , μ_n , σ_c , σ_n from equations (2.6), (2.8), (2.7) and (2.9) respectively and solve for p , we get $p = 414$.

If we threshold the sums by this decision boundary, we will be losing some information at the same time removing some noise. The subthreshold sums are reset to 0 and superthreshold sums are set to 1. This results in two types of error: *dropped* and *residual*.

The probability of losing information in the coactive region will be the probability of the sum being subthreshold. Let us refer to these bits as “*dropped bits*”.

$$P [\text{dropped bits}] = P_d = \sum_{i=0}^{d'} \binom{p}{i} (q_c)^i (1 - q_c)^{p-i} \quad (2.13)$$

where d' is the nearest integer to d .

Some noise in non-active region is removed. Still there are some bits above threshold which will cause error. Let us refer to these bits as “*residual bits*”.

$$P [\text{residual bits}] = P_r = 1 - \sum_{i=0}^{d'} \binom{p}{i} (q_n)^i (1 - q_n)^{p-i} \quad (2.14)$$

If all the bits corresponding to one mitral patch are added, we will get some distribution of the sum.

In non-active region, there will be only residual bits. Let's denote the sum in the non-coactive region by S_n and number of levels in one thermometer by n . We can write,

$$P[S_n=i] = \binom{n}{i} (P_r)^i (1 - P_r)^{n-i} \quad (2.15)$$

Expected value and variance of S_n are given by

$$\mu_{S_n} = nP_r \text{ and } \sigma_{S_n}^2 = nP_r(1 - P_r) \quad (2.16)$$

respectively.

Let's denote the sum in active region by S_a . We can write

$$P[S_a=i] = \binom{n}{i} [P_c(1 - P_d) + P_n P_r]^i [1 - P_c(1 - P_d) - P_n P_r]^{n-i} \quad (2.17)$$

The expected value and the variance of S_a are then given by

$$\mu_{S_a} = n[P_c(1 - P_d) + P_n P_r] \text{ and} \quad (2.18)$$

$$\sigma_{S_a}^2 = n[P_c(1 - P_d) + P_n P_r][1 - P_c(1 - P_d) - P_n P_r] \quad (2.19)$$

respectively.

Now again we have to decide some decision boundary between μ_{S_a} and μ_{S_n} . If we consider σ_{S_a} and σ_{S_n} approximately equal, the decision boundary is given by

$$d_s = \frac{\mu_{S_a} + \mu_{S_n}}{2} + \frac{\sigma_{S_n} \sigma_{S_a}}{\mu_{S_a} - \mu_{S_n}} \ln \left(\frac{P_{S_n}}{P_{S_a}} \right) \quad (2.20)$$

where P_{S_n} and P_{S_a} is the probabilities of occurrence of sums in non-active and active regions respectively.

If we threshold the sums using this decision boundary and reset the sub-threshold sums to zero, we can calculate the expected value and the variance of the new distribution similarly. Let's denote the expected value and the standard deviation of the sum in the

active region after thresholding by μ_{Sa}' and σ_{Sa}' respectively and those in non-active region by μ_{Sn}' and σ_{Sn}' respectively.

Now we can write the signal to noise ratio as:

$$SNR = \frac{\mu_{Sa}'}{\sqrt{\mu_{Sn}'^2 + \sigma_{Sn}'^2 + \sigma_{Sa}'^2}} \quad (2.21)$$

For the set of parameters described above, ($p = 30$, $h = 16$) we get following results analytically:

$$P_d = 0.48 \quad P_r = 0.31$$

$$\mu_{Sa} = 6.64 \quad \sigma_{Sa} = 1.97$$

$$\mu_{Sn} = 4.96 \quad \sigma_{Sn} = 1.85$$

$$d_s = 6.75$$

$$\mu_{Sa}' = 5.62 \quad \sigma_{Sa}' = 3.43$$

$$\mu_{Sn}' = 2.58 \quad \sigma_{Sn}' = 3.39$$

$$SNR = 1.0277 = 0.2375 \text{ dB}$$

So we conclude that backpropagation through the naive network will not produce reliable results.

2.11 LVQ Network

To combine the clusters identified by the proposed model into a complex cluster the network can be extended to LVQ network (refer section 1.5.4) by adding one more layer of network. An LVQ network is a two layer network. The first layer of the LVQ network is a competitive layer. The second layer of the LVQ network is used to combine different subclasses identified by the first layer into a class. This layer can be as simple as a programmable logic array (PLA).

Let us assume that the clusters created by our network are subclusters. We can form clusters of different sizes and shapes if we could somehow combine these clusters. A PLA can serve this purpose.

Fig. 23 shows the block diagram of the LVQ layer. The input to the LVQ layer is the output of the piriform (or WTA) patches which is a binary string. Since only one bit out of h bits corresponding to a piriform patch can be active at any time, it is advisable to use a binary encoder that can reduce the number of bits to $\log_2 h$ where h is the number of bits in one patch. Let the output of the binary encoder be $B=[B_1 B_2 B_3 \dots B_v]$, where $v=p\log_2 h$.

The output of the binary encoder is fed to the PLA. There are two programmable arrays, AND array and OR array.

The input of each AND gate is the binary code of some subcluster. There can be 2^v such codes. Let u be the actual number of AND gates (or the codes used) out of 2^v . Let us denote the outputs of the AND gates by $A=[A_1 A_2 A_3 \dots A_u]$.

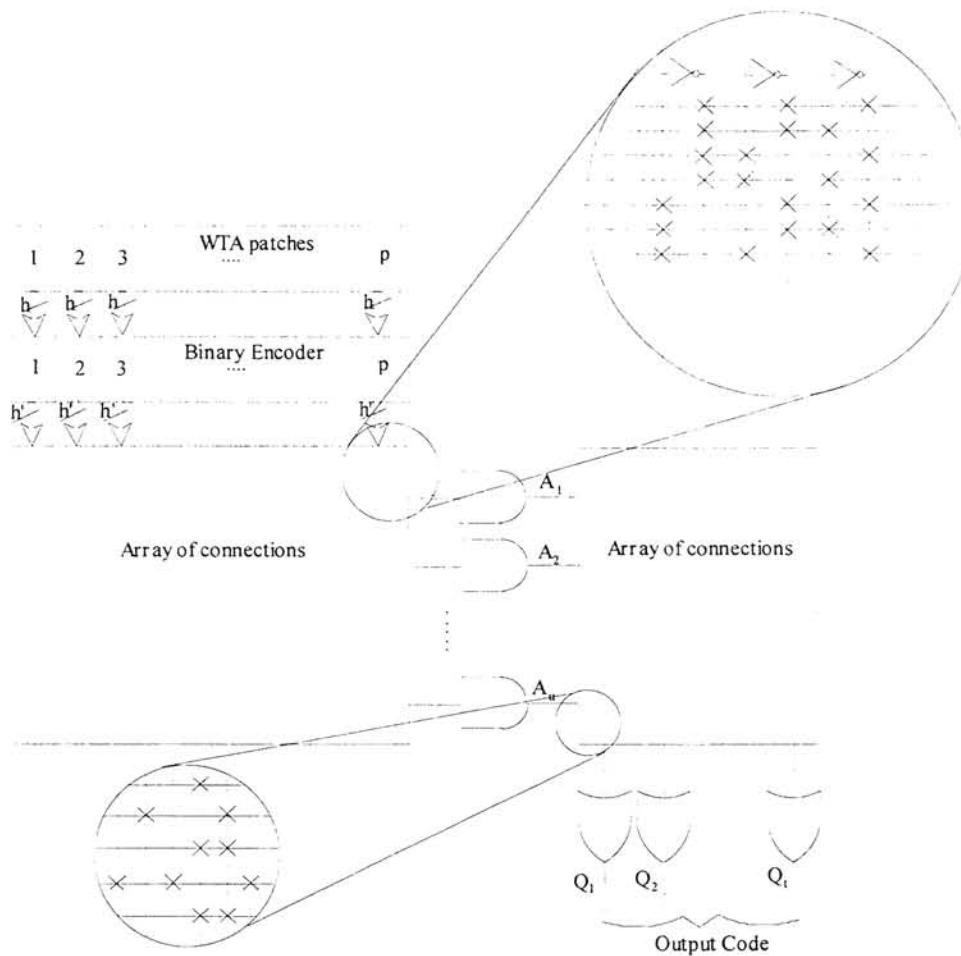


Fig. 23 LVQ layer : Block diagram

Let the AND array be denoted by R . R can be written as follows:

$$R = \begin{bmatrix} R_{110} & R_{111} & R_{120} & R_{121} & \dots & R_{1v0} & R_{1v1} \\ R_{210} & R_{211} & R_{220} & R_{221} & \dots & R_{2v0} & R_{2v1} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ R_{u10} & R_{u11} & R_{u20} & R_{u21} & \dots & R_{uv0} & R_{uv1} \end{bmatrix}$$

The elements of R are either 1 or 0 but R_{xy0} and R_{xy1} both cannot be 1 at the same time.

Now the output of the x th AND gate can be written as

$$A_x = (\overline{B_1}R_{110} + B_1R_{111}) \cdot (\overline{B_2}R_{x20} + B_2R_{x21}) \cdot \dots \cdot (\overline{B_v}R_{xv0} + B_vR_{xv1}) \quad (2.22)$$

When all the inputs of the AND gate are asserted, the output of the AND gate will enable a corresponding **row** in the OR array. According to the bits stored in the OR array, network output appears at the output of the OR gates. Let the output code consist of t bits. In other words, there are t bits stored in each **row** of the OR array. Let the bits stored in the OR array be denoted by O .

$$O = \begin{bmatrix} O_{11} & O_{12} & \dots & O_{1t} \\ O_{21} & O_{22} & \dots & O_{2t} \\ \vdots & \vdots & \ddots & \vdots \\ O_{u1} & O_{u2} & \dots & O_{ut} \end{bmatrix}$$

Let us denote output code by Q , where

$$Q = [Q_1 \ Q_2 \ \dots \ Q_t]$$

Now we can write Q_y as

$$Q_y = A_1O_{1y} + A_2O_{2y} + \dots + A_uO_{uy}$$

So when output of x th AND gate goes high, x th **row** of the matrix O , denoted by O_x , will appear at the output since all other AND gates have low output.

Now the question is how do we combine different clusters. Let us assume that we want to combine cluster x and cluster y . If they are combined they should produce the

same output code. This can be achieved by making row O_x and row O_y of the matrix O identical.

By extending the network to LVQ network, we can form clusters of various shapes and sizes on the hypersurface. By forming concave surfaces, we can solve linearly inseparable problems.

CHAPTER 3

CONCLUSIONS AND FUTURE PROSPECTS

A new model, which is a modified version of the Coultrip-Granger model, is proposed. The goals while developing the new model were to retain the desirable properties of the original model including hierarchical clustering, fast and rapid learning, a hardware compatible implementation and to control the capture mechanism. We have also addressed the issue of a dimensioning methodology to determine the dimensions of the weight matrix. An attempt was made to reconstruct the template by backpropagation of the output code through the naive weight matrix. Finally we have introduced an additional layer to the network which converts the proposed network into an LVQ equivalent. The LVQ network has the capability of forming concave clusters and thus solving linearly inseparable problems.

Simulation results have shown that the proposed model clusters the input data. Since the basic learning rule is the LTP rule with coarse weight increments, the property of rapid learning is maintained. Also the proposed training algorithm ensures that there is

no undesired capture by precedence or capture by overtake. Unlike the Coultrip-Granger model that requires the class information *a priori*, the proposed model does not require the class information. Also by making the two pathways exactly identical, it becomes easier to implement these weight matrices electronically. In the proposed model, the user has to define the cluster size, thus has direct control over the quantization factor in sampling the hypersurface.

It has been identified that the size of the input vector (LOT) cannot be increased indefinitely (Refer to section 2.4.1). Still we have no control over LOT dimension since the number of inputs is application specific whereas the number of mitral cells in each mitral patch (or the number of levels in the thermometer code) is determined by the fidelity requirement. The number of LOT lines is a product of the number of inputs and the number of levels in the thermometer code. However if the dimension of LOT vector is too large, one can adjust the weight increment (Δw) accordingly (Refer to section 2.4.1). The number of cells in a piriform patch should be moderate in order to have less overlap between the output codes of the patterns having less overlap with each other. A patch size of 16 is a good choice since a further increase in patch size offers little benefit, i.e. the expected number of active synapses on the winner cell does not increase considerably. Also for reconstructing the template with maximum fidelity, it is necessary to have appropriate number of piriform patches. Also a proper choice of number of piriform patches ensures a sufficient hamming distance between the output codes of the clusters.

These four items (number of input signals, fidelity requirement, piriform patch size and number of piriform patches) determine the dimensions of the weight matrix.

It has been shown that backpropagation through the naive or fixed weight matrix will not produce reliable inhibitory feedback. But a heuristic approach, that exploits the knowledge of the structure of the thermometer code but not mentioned in this thesis due to lack of strong theoretical background, shows some promise in reconstructing the template.

It has been shown that a PLA can be added to the network to achieve an LVQ network. Thus we can form complex clusters of various shapes, including concave. The LVQ network can solve linearly inseparable problems.

Let us consider a practical example of an ECG classifier. Let us assume that applied time series consists of 250 samples starting at QRS minus fifty and the number of possible ECG classes is limited to less than 30 while the fidelity requirement is 44 dB.

The 44 dB fidelity requirement sets the number of levels in the thermometer code (number of mitral cells per patch) to 128. The number of inputs is 250, hence 250 mitral patches are required. These two factors alone determine the number of LOT lines, i.e. $N=250 \times 128$. From section 2.2, we know that the number of piriform cells per piriform patch should be of moderate size with h equal 16 being an acceptable choice.

The factors to be determined are the number of piriform patches and the hamming distance between two classes. Since the number of possible classes is limited to 30, assuming that the clusters are distributed uniformly in the input space, we can determine

an average hamming distance between the classes. But for determining the number of piriform patches, there is no known optimum solution. Thirty patches is a good starting point since it ensures that the dropped bit error while reconstructing the template is negligible. After a number of iterations we can arrive at an acceptable solution. If the number of patches required are more than the actual number of patches fabricated on an IC, two or more ICs can be connected in parallel. But if the application requires larger N than fabricated, we cannot connect ICs in order to achieve larger N . Further research is needed to determine an optimum size of the weight matrix. The proposed model has the following limitations and further research in those areas is needed.

Even though the simulation results show clustering properties of the network, no attempt is made to show hierarchical clustering. Before hierarchical clustering we must ensure that the template is reconstructed with sufficient fidelity. In the proposed training algorithm, the user has control over the cluster size, i.e. he has control over the quantization with which the hypersurface is sampled, although how to use this effectively is still to be discovered. Taking into consideration the large number of possible output codes, it is impossible to implement a PLA that includes all these combinations. Some statistical experiments should be carried out in order to optimize the size of the AND array. Also training algorithm for the PLA needs to be developed.

REFERENCES

1. Zurada, "Introduction to Artificial Neural Networks," West Publishing Company, 1992.
2. M. T. Hagan, M. Beale, H. B. Demuth, "Neural Network Design," 1995.
3. M. Ramanathan, "Statistical Modeling of an Electronic Olfactory," Master Thesis, OSU, 1995.
4. S. Patil, "VLSI Implementation of Olfactory Cortex Model," Master Thesis, Oklahoma State University, 1992
5. J. Choi, B. L. Sheu, "A High-Precision VLSI Winner-Take-All Circuit for Self-Organizing Neural Networks," IEEE Journal of Solid State Circuits, Vol. 28, No. 4, pp. 576-583, 1993.
6. K. Wagner, T. M. Slagle, "Optical Competitive Learning with VLSI Liquid Crystal Winner-Take-All Modulators," Applied Optics, Vol. 32, No. 8, pp. 1408-1435, 1993.
7. C. A. Mead, X. Arreguit, J. Lazzaro, "Analog VLSI Model of Binaural Hearing," IEEE Transactiona on Neural Networks, Vol. 2, pp. 230-236, 1991
8. S. V. Gala, "System Level Integration of Olfactory Cortex Model," Master Thesis, Oklahoma State University, 1993.
9. C. A. Mead, Mahowald, "A Silicon Model of Early Visual Processing," Neural Networks, Vol. 1, pp. 91-97, 1988.
10. R. Coultrip, R. Granger, "Sparse Random Networks with LTP Learning Rule Approximate Bays Classifier Via Parzen's Method," Neural Networks, Vol. 7, No. 3, pp. 463-476, 1994.
11. B. Everitt, "Clustering Analysis," Third Edition, Edward Arnold, 1993.
12. P. Shoemaker, "Analysis of Elementary Clustering Mechanisms in a Model of Early Olfactory Processing," 1995.

13. J. Ambros-Ingerson, R. Granger, G. Lynch, "Simulation of Paleocortex Performs Hierarchical Clustering," *Science*, Vol. 247, pp. 1344-1348, 1990.
14. J. Ambros-Ingerson, "Computational Properties and Behavioral Expression of Cortex-Peripheral Interactions Suggested by a Model of the Olfactory Bulb and Corex," Ph.D. Dissertation, University of California, Irvine, 1990.
15. G. Lynch, R. Granger, "Simulation and Analysis of a Simple Cortical Network," *Psychology of Learning and Motivation*, Vol. 23, pp. 205-241, 1989.
16. J. G. Taylor, "Model of Olfactory Learning," Department of Mathematics, King's College, London, pp. 277-280.
17. Z. Li, J. Hopfield, "Modeling the Olfactory Bulb and its Neural Oscillatory Processing," *Biological Cybernetics*, Vol. 61, pp. 379-392, 1989.
18. Z. Li, "A Model of Olfactory Adaptation and Sensitivity Enhancement in The Olfactory Bulb," *Biological Cybernetics*, Vol. 62, pp. 349-361, 1989.
19. C. Linster, C. Masson, M.Kerszberg, L. Personnaz, G. Dreyfus, "Computational Diversity in the Formal Model of the Insect Olfactory Macrogglomerulus," *Neural Computation*, Vol. 5, pp. 228-241, 1993.
20. M. Schwartz, "Information Transmission Modulation, and Noise," Second Edition, McGraw-Hill Book Company, 1970.
21. J. E. Freund, "Mathematical Statistics," Fifth Edition, Prentice Hall, Englewood Cliffs, 1992.

VITA

RAHUL RATILAL SHAH

Candidate for the degree of

Master of Science

Thesis: A STATISTICAL MODELING OF AN OLFACTORY SYSTEM FOR
ELECTRONIC IMPLEMENTATION

Major Field: Electrical Engineering

Biographical:

Personal Data: Born in Pune, India, on October 19, 1971, son of Ratilal and Bharati Shah

Education: Graduated from Sir Parashurambhau College, Pune, 1989; received Bachelor of Engineering degree in Electronics Engineering from University of Poona, Pune, India in June 1993; completed requirements for the Master of Science degree at Oklahoma State University in July 1996.

Experience: Research Assistant (January 1995 to present) Department of Electrical and Computer Engineering, OSU; worked at NRaD, San Diego, CA on Master's Thesis

Teaching Assistant (August 1995 to present) Department of Electrical and Computer Engineering, OSU.