PERFORMANCE INVESTIGATION OF ARTIFICIAL NEURAL NETWORK MODELS OF ASSOCIATIVE MEMORIES

BY

JUN LIU

A Thesis

Submitted to the Faculty of Graduate Studies

in Partial Fulfillment of the Requirements

for the Degree of

MASTER OF SCIENCE

Department of Electrical and Computer Engineering

University of Manitoba

Winnipeg, Manitoba

ISBN 0-315-77936-5

Canada

PERFORMANCE INVESTIGATION OF ARTIFICIAL NEURAL NETWORK MODELS OF ASSOCIATIVE MEMORIES

BY

JUN LIU

A Thesis submitted to the Faculty of Graduate Studies of the University of Manitoba in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

ABSTRACT

The work described in this thesis is motivated by the need to understand the behavior of neural network models used as associative memories. Part of this work is devoted to the study and design of a way of performance evaluation which can sufficiently evaluate the main properties of associative memory, effectively be implemented in the experiment, and achieve highly reliable results. The procedure developed for testing the memory performances follows the black box strategy. This procedure has been uniformly applied to investigating both unidirectional and bidirectional memories. The results of each well–defined performance characteristics for the Hopfield network, bidirectional associative memory, Ho–Kashyap encoded memory, and Backpropagation network are presented. These results show that the quality of accretive recall is affected by the dimension of the input pattern. The bidirectional search can improve the accretive recall in some cases, but it may also deteriorate the interpolative recall. The maximum capacity for the Ho–Kashyap model, with the same dimension in its input and output patterns, is approximately equal to 1.4 times the dimension of input or output pattern. Results also indicate that the Ho–Kashyap model may be best–suited in performing accretive recall while the backpropagation model is very good at realizing interpolative recall. The investigation verifies that the capacity of bidirectional associative memory is much lower than Kosko's original estimation. The value of $n/2\log_2 n$, which was analytically derived by McEliece, et al. to estimate the asymptotic capacity of a Hopfield network with n neurons [MPRV1987], can be directly applied to estimating the capacity of the bidirectional associative memory.

ACKNOWLEDGEMENTS

Certain individuals who contributed in their own ways to this study deserve to be acknowledged. First and foremost, heartfelt thanks to Dr. Witold Pedrycz, who not only provided guidance but much encouragement and support. His background ideas, valuable comments and thoughtful criticism were more helpful than he will know.

Special thanks are given to fellow graduate student, Qinyan Dai, who responded willingly to answer my endless questions. He was also helpful in his technical expertise and offered precise suggestions for the improvement of this thesis.

There are other individuals to whom I happily extend my sincere thanks for their assistance. I am grateful to Erwin Dirks and Ken Ferens who contributed their valuable time to read this thesis. Their suggestions and comments are much appreciated. I am also thankful to Liz Vensel who devoted her sparetime in improving my written English.

Finally, sincerest thanks are reserved for my wife Min Xia, my sister Qi Liu, and my little son Hao Liu. Had it not been for their love, understanding and support, I would never have completed this study.

TABLE OF CONTENTS

CHAPTER 5 CONCLUSIONS AND RECOMMENDATIONS

LIST OF FIGURES

Figure **Page**

Figure **Page**

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

Symbol	Denotes
$A_{P_e}^{p_k}$ and $B_{P_e}^{p_k}$	Information capacity measured under specified conditions (see section 2.2.3 in chapter 2)
C_{ap}	Information capacity
C_p	Number of trials for a particular pre–defined noise probability P_e
D_H	Hamming distance
E	Energy function or cost function
Hd	The maximum mutual Hamming distance in the patterns subject to that the specified number of patterns, p', that can be generated
k	The number of hidden neurons
m	The dimension of output pattern
MD	The value of minimum distance
n	The dimension of input pattern
$O(.)$	Computational complexity of an encoding or recall algorithm
p	The number of stored patterns/associations in a particular training set
p'	The number of patterns can be generated under the condition that the maximum mutual Hamming distance in the patterns is Hd
$PCSS$	Probability of memory converging to spurious states
$PCOS$	Probability of memory converging to oscillatory states
P_e	Probability of each bit being reversed in input patterns
p_k	The k–th training set

$p_{\max}$	The maximum number of training patterns/associations required in estimating information capacity
P_r	Probability
t	Iterative number
T_p	The number of groups of training sets used in the testing (each training set contains p pairs of patterns/associations)
$T(.)$	Transformation function
$\mathbf{W}$	Weight matrix (m by n), $\mathbf{W} = [\, w_{ji} \,]$, j=1, 2, ..., m, i=1, 2, ..., n
$\mathbf{x}^{(s)}$	Input (training) vector/pattern, $\mathbf{x}^{(s)} = (x_1^{(s)}, x_2^{(s)}, \ldots, x_n^{(s)})^T$, s=1, 2, ..., p
$\mathbf{x}^{(init)}$	Initial state
$\mathbf{X}$	Input matrix (n by p), $\mathbf{X} = [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(p)}]$
$\mathbf{y}^{(s)}$	Output (training) vector/pattern, $\mathbf{y}^{(s)} = (y_1^{(s)}, y_2^{(s)}, \ldots, y_m^{(s)})^T$, s=1, 2, ..., p
$\mathbf{Y}$	Output matrix (m by p), $\mathbf{Y} = [\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \ldots, \mathbf{y}^{(p)}]$
$\mathbf{y}^{(recall)}$	Recalled pattern
Z_{AB}	The effect of input and output pattern dimensions on information capacity
α	Momentum coefficient
γ_i	Threshold in the neuron which receives i–th component in the input pattern (Note: γ_i has no effect on the input pattern but only on the signal sent by other neurons)
θ_j	Threshold in output neuron j
η	Learning rate
ϵ	Noise component in an input pattern

CHAPTER 1

INTRODUCTION

1.1 Motivation and Purpose

This thesis is motivated by the recent renewed interest in the artificial neural network model of associative memories. Recent studies of neuroanatomical brain functions have provided fertile ground for the development of neural network models of associative memories (referred to as associative memories) [HiAn1989]. One possible reason for the rapid growth in this area may be due to the fact that, although traditional digital computers based on the localized principle have achieved considerable success in many areas, there still exists a class of problems which seems to be very difficult for normal computers to solve. An example of such a task is retrieving an item or an association when given an incorrect or partial description of its features. While this task does not appear to be difficult for humans to perform, it is inappropriate for the conventional computer. The reason for the difficulty is that the digital machine accesses items in its memory by using their addresses, and it is hard to discover the location of an item from an arbitrary subset of its contents. Furthermore, for some tasks such as identifying handwritten characters, understanding continuous speech, and solving complex pattern recognition problems, it is very difficult to provide step by step procedures for a conventional computer to follow, and, therefore, it is very impractical to solve these problems through procedure–based systems.

The fact that biological memories are so effective in undertaking certain tasks suggests that it may be possible to obtain similar capabilities in artificial devices based on the design

principles of biological neural systems. For many years, researchers have developed many types of associative memories. These memory models bear a resemblance to the human brain in the sense that: (i) memory typically consists of densely interconnected processing elements, (ii) knowledge is acquired through training (rather than programming) and is retained in the strength of interconnections among processing elements, and (iii) knowledge stored in the memory takes the form of a stable state (rather than in a particular location as in normal computers).

Although today there exist many associative memory models, their behavior has not been adequately captured. One reason for being unable to obtain the behavior is the lack of clear definitions of the performances. Another reason is the lack of a well–developed methodology which can be used to extract the interesting properties in these artificial devices. One purpose of this thesis is to present clear statements to describe memory characteristics. These characteristics are not only defined literally but also formulated mathematically. These definitions help one to arrive at quantitative descriptions of memory performances. The other goal of this research is to provide a systematic strategy of investigation. This strategy is based on probabilistic and statistical theories. The uniform treatment in the investigation of memory models makes comparison of memory performance possible. This thesis will also provide the results from investigating four different types of associative memories. These results can be regard as the complementary source of the information that guide one towards the goal of understanding the behavior of these newly born intelligent devices.

1.2 Localized and Distributed Memory Storage

Two contrasting ideas can be identified in the history of brain science concerning where and how information is stored [Aqui1987]. One traditional view is on the localization and

determination concepts forcefully advocated by many neurobiologists and psychologists [Hebb1960], [Kand1976]. In their view, the brain system is made up of identifiable, localized parts, and behavioral functions can be localized to particular components. Under this assumption, "there seems to exist a coarse specialization of the brain areas according to the various sensory modalities (visual, auditory, somatosensory, etc.) as well as different levels of operations (speech, planning of actions, etc.)" [Koho1984]. (See Fig. 1.1).

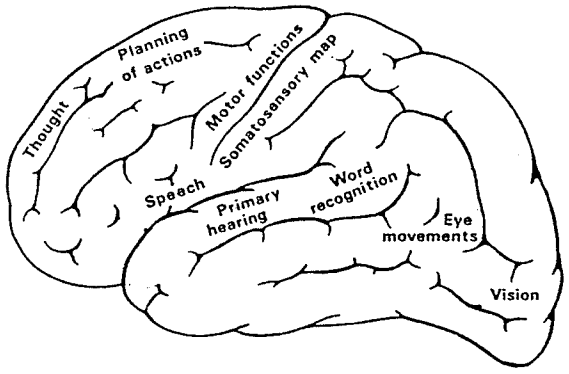

Fig. 1.1 Brain areas. [Koho1984] Copyright by Springer–Verlag Berlin Heidelberg 1984.

This school of thought persisted in the experimental work of P. P. Broca, A. R. Luria and L. R. Aquire [Broc1960], [Lura1966], [Aqui1987] and was believed to be the theoretical foundation for the invention of powerful digital computers.

The other viewpoint arose in opposition to the traditional view and was developed out of the localized principle. In this principle, the behavior and mental activity result from the integrated activity of the entire brain [RuOr1977], [Aqui1987]. The idea is that memory involves a constant change in the relationship among all neurons. This kind of change is accomplished through either structural modifications or biochemical events within neurons in such a way that neighboring neurons communicate. In this view, information is not located

in any particular place, but is stored in the relationship among neurons which participate in the encoding of information.

1 .3 The Role and the Structure of Associative Memory

1 .3 .1 Associations

One of the basic elements of human memory is an association [Koho1977]. Essentially, an association is a rule or a relationship between two stored memory traces that map one into the other. In a library card index, you can find the title of a book if given the author's name. If you know a key word, you can use the subject index instead. In the same way, by picturing the face of a friend, you can remember his name. So there is an association between name and the two stored pieces of information in a memory system. Mathematically, if two stored pieces of information can be properly represented in a vector format as $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$ and $\mathbf{y} = (y_1, y_2, \ldots, y_m)^T$, where $\mathbf{x} \in R^n$ and $\mathbf{y} \in R^m$, and if $T(.)$ is the function mapping $\mathbf{x}$ into/onto $\mathbf{y}$ in the expression $\mathbf{y} = T(\mathbf{x})$, then $T(.)$ is an association.

At a very general level, the role of associative memory in the information processing account of cognitive behavior is that when a member of an associated pattern $\mathbf{x}$, which is considered as a "key", is entered into a memory system, the memory gives output $\mathbf{y}$ which is related to the "key". The memory process which transforms the ordered set of input patterns $\mathbf{X}$ into the other set of output patterns $\mathbf{Y}$ can be schematically represented by Fig. 1.2 or Fig. 1.3 depending on what the type of memory storage method is employed (the localized or distributed). The system depicted by the flow chart in Fig. 1.2 is obviously dedicated to a localized memory model. This flow chart is to be read from left to right.

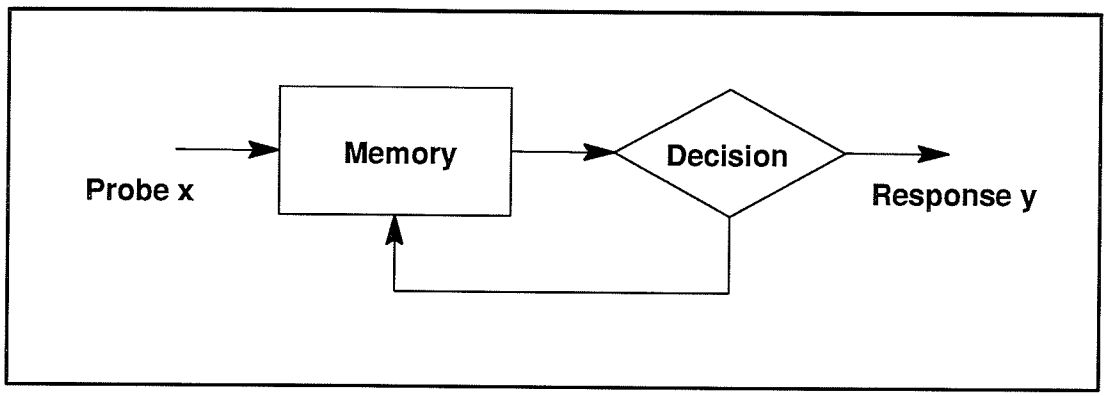

Fig. 1.2 A simple flow chart representing the memory process performed in localized memories.

The probe **x** stands for a search argument or a feature vector and the arrow leading from the probe **x** into the box labeled memory indicates that the physical pattern is entered and interrogated. The comprehensive and transforming process which is represented by the memory–decision–response loop is actually the searching–matching process realized by means of executing a pre–defined program. The stop rule which is not represented in this simple flow chart is based on some similarity measures. An example of this device is the conventional data base system designed to handle relational structures by means of both complex data structures and a pre–programmed search algorithm.

Memories based on localized information storage and retrieval mechanisms are contrasted to distributed memories model in both structure and manner of processing. As illustrated in Fig. 1.3, the associative memory consists of a number of neurons. These neurons are densely interconnected with each other through linear connections called weights. The input/output relationship is described by neuron's transform functions. One of the common transform functions is

$$y_j = f\left(\sum_{i=1}^{n} w_{ji}\, x_i - \theta_j\right) \tag{1.1}$$

where w_{ji} is the weight from input neuron i to output neuron j, θ_j is the threshold in neuron j and f(.) is a nonlinear activation function. Typical activation functions are the hardlimiter, unit step, and sigmoid which are shown in Fig. 1.4. In this type of memory model, all the values in the universe of variables (weights and thresholds) which satisfy the condition of associative mapping constitute possible solutions. In general, the distributed memory takes the relationships of internal processing elements (or actually their strength) into account statistically [Koho1984].

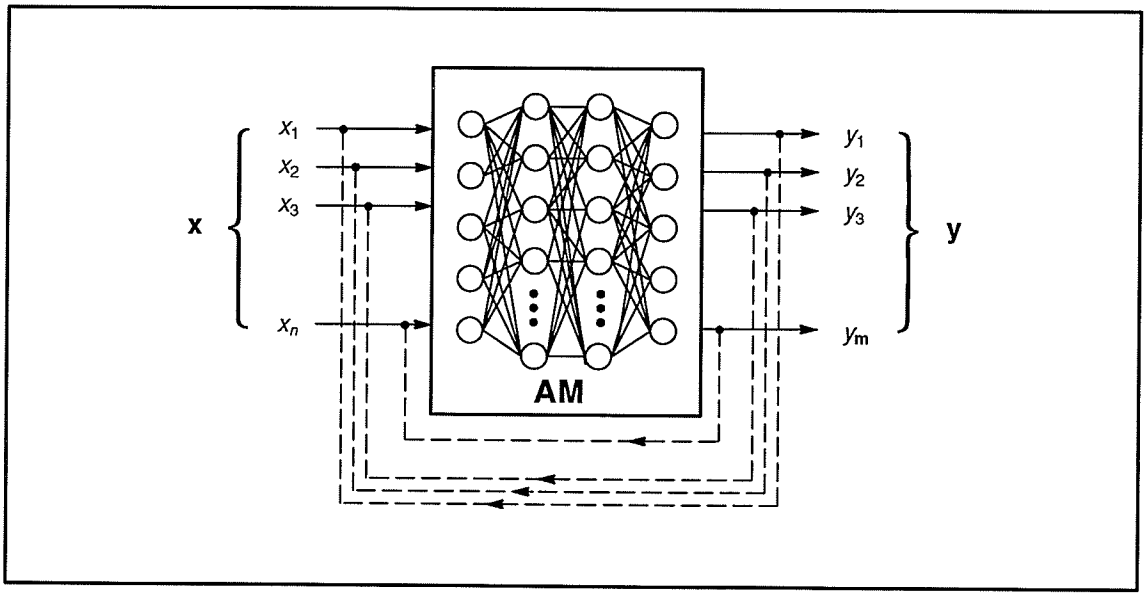

Fig. 1.3 A simplified artificial neural network model of associative memories.

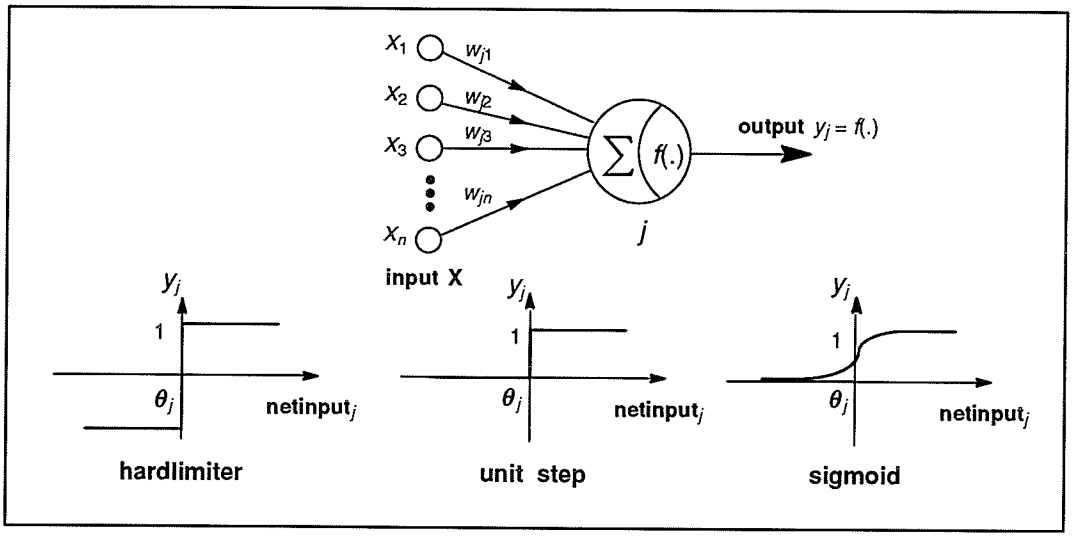

Fig. 1.4 Artificial neuron

1 .3 .2 Associative recall

A central operation in explaining the function of associative memories is called *associative recall.* The associative recall is defined as any process that when given an input, the memory is able to evoke a specific response in a highly selective fashion associated with that input [Koho1984]. Under this definition, the associative recall can be simply described as an optimal mapping from a set of input patterns $\mathbf{x}^{(s)}$, s=1, 2, ..., p, into/onto a set of output patterns $\mathbf{y}^{(s)}$, s=1, 2, ..., p. The optimal mapping means when a memory is evoked by an input pattern, $\mathbf{x}'$, the recalled pattern $\mathbf{y}'$ has the properties that $\mathbf{y}'$ ultimately matches with $\mathbf{y}^{(k)}$ (which is subset of $\{\mathbf{y}^{(s)}\}$, s=1, 2, ..., p) if $\mathbf{x}^{(k)}$, $k \in \{1, 2, \ldots, p\}$, is the best pattern (in the subset of $\mathbf{x}^{(s)}$ s=1, 2, ..., p) that represents the input pattern $\mathbf{x}'$.

There are two alternative ways in performing associative recall. Components in the network can either change their states one at a time, which is referred to as *asynchronous* recall, or change all at once, which is called *synchronous* recall. The memory that utilizes

asynchronously recall strategy is usually called an asynchronous model. Similarly, the memory that adopts synchronously recall strategy is referred to as a synchronous model. The distinction between these two models lies in the fact that a particular component in the asynchronous model is modified to provide some shorterm effect on previous states which may immediately affect other neurons, whereas the synchronous model only collects previous states and generates new states as a whole.

1 .3 .3 The structure of associative memory

In terms of information processing, existing memory models can be categorized into two different types. One is *feedforward associative memory* [HiAn1989], the other is *dynamic associative memory* [Hassoun1989]. The broken lines in Fig. 1.3 are to indicate that for associative memories, there may or may not be a signal feedback from their output to input. If an associative memory has no feedback loop, the system is referred to as a feedforward associative memory, otherwise it is named as a dynamic associative memory. Neurons in feedforward memory only propagate information from input neurons to output neurons through hidden neurons if they exist. The number of hidden layers and the number of neurons in each layer in the memory are problem dependent. For dynamic memories, neurons change their activation iteratively in the process of associative recall. Such a dynamic process terminates only if the memory settles down in one of the stable states.

Further distinctions can be made according to the input and output dimensions. An associative memory is defined as an *autoassociative memory* [Koho1984] if its input dimension n equals the output dimension m ($n=m$). This type of memory is dedicated to the reconstruction of the pattern stored in memory if the memory is evoked by noise corrupted or partially absent inputs. Thus, both the input and output patterns being processed

in the autoassociative memory belong to the same vector space. The other type of associative memory, called *heteroassociative memory* [Koho1984], is designed to perform an associative mapping from one space into/onto another. Two sets of patterns in the heteroassociative memory can be selected freely and independently and they distribute in two different vector spaces. Thus, the heteroassociative memory differing from the autoassociative memory in terms of topological structure is that the dimensions of input and output for heteroassociative memories are not identical, i.e. $n \neq m$.

1.4 Thesis Organization

This chapter serves as an introduction to the entire thesis. The purpose of this chapter is to provide the fundamental concept of localized and distributed memory models as well as to outline their major differences in encoding, storing and retrieving mechanisms. Chapter 2 characterizes the main features that reflect the behavior of associative memories. The definition for each feature is given in the context followed by a discussion and explanation. These features are mathematically formulated so that they are readily applied to performance testing. The necessity, sufficiency and validity of the minimum distance constraint used in testing the memory performance is thoroughly discussed. Elements for performance testing, such as the formation of training and testing patterns are provided. A procedure used for testing memory performances is derived. Chapter 3 is divided into four sections. Each section begins with the general review of the memory model being tested followed by experimental results. Descriptions and analysis of these results are given at the end of each section. Chapter 4 compares the performance of the models tested in chapter 3. The comparison is based on the following aspects: (i) information capacity, (ii) error correction capability, (iii) the effect of input and output pattern dimensions on accretive recall, (iv) the probability of the memory getting into false states, and (v) the temporal

complexity of encoding and associative recall. Conclusions and recommendations are given in Chapter 5.

1.5 References

[Aqui1987] Aquire, L. R. (1987), "Memory and Brain", Oxford University Press, 1987.

[Broc1960] Broca, P. P. (1960), "Remarks on the seat of faculty of articulated language, followed by an observation of aphemia", Translated by G. V. Bonin, In: Some papers on the cerebral Cortex, Springfield, IL: Thomas.

[Hass1989] Hassoun, M. H. (1989), "Dynamic heteroassociative memories", Neural Networks, Vol. 2, pp. 275–287.

[Hebb1960] Hebb, D. O. (1960), "The organization of behavior", New York: Wiley.

[HiAn1989] Hinton, G. E. and Anderson, J. A. (Eds.) (1989), "Parallel Models of Associative Memory", Updated Edition, Hillsdale, N. J. : Lawrence Erlbaum Associates.

[Kand1976] Kandel, E. R. (1976), "Cellular basis of behavior", San Francisco: Freeman.

[Koho1977] Kohonen, T. (1977), "Associative Memory, a System–Theoretical Approach", Berlin: Springer–Verlag.

[Koho1984] Kohonen, T. (1984), "Self–organization and Associative Memory", Berlin: Springer–Verlag.

[Kosk1988] Kosko, B. (1988), "Bidirectional associative memories", IEEE Transactions on System, Man and Cybernetics, Vol. 18, No. 1, pp. 49–60.

[Kron1986] Kronsjo, L. (1986), "Algorithm: the Complexity and Efficiency", Second Edition. John Wiley & Sons.

[Lura1966] Luria, A. R. (1966), "Higher Cortical Functions in Man", London, Tavistock.

[RuOr1977] Rumelhart, D. E. and Ortony, A. (1977), "The representation of knowledge in memory", In R. C. Anderson, R. J. Spiro and W. E. Montague (Eds.), Schooling and the acquisition of knowledge, Hillsdale, N. J. : Lawrence Erlbaum Associates.

[RuMc1986] Rumelhart, D. E., McCelland, J. L and the PDP Research Group, (1986), "Parallel Distribution Processing: Exploration in the Microstructure of Cognition, Vol. 1, Foundations", The MIT Press Cambridge, Massachusetts.

CHAPTER 2

METHOD FOR PERFORMANCE EVALUATION

2. 1 Introduction

This chapter sets forth a procedure for performance evaluation for the purpose of discovering the behavior of the neural network model of associative memory. The method, as presented here, for testing memory performances is a general one in the sense that it is not based on any particular task. Implementing performance evaluation requires two sets of data: the training patterns/associations and the testing patterns. Training patterns/associations represent the information to be stored in the memory. Testing patterns serve as *keys* which are used to stimulate the memory. To ensure generality, these data should be randomly generated from a uniform distribution. The method used for testing memory performances follows a *black box* approach. Based on the theory of the black box, the entire memory is treated as a completely unknown system. Neither the internal structure nor the internal activation of a memory is considered in the testing except for the memory response to the environment.

The primary reasons for adopting the black box testing approach are that: firstly, theoretical analysis of memory performance, such as, accuracy, capacity and achievable resolution still remains in a state of infancy, since it involves a lot of assumptions and unrealistic simplifications. Secondly, although the hypothesis that simple networks behave as if they minimized the quantity of the energy in a physical system has proved to be a very

useful tool in expressing nonlinear cross–coupled networks, it is still very difficult to apply to an arbitrary network model. This is because, firstly, for most associative memories, there is no guarantee that memory will settle down in the nearest energy minimum and few patterns can be stored without creating spurious local minima. Secondly, for some learning algorithms, such as the trial–and–error process, weights are not only the function of training patterns but also the function of weights themselves. As a result, it seems to be impossible to write down the relationship between the memory response and the formation of weights in either a closed or series form. These relationships, however, are indispensable in obtaining exact memory performances analytically. For these reasons, treating a whole system as a black box is an alternative approach to gain the behavior of associative memory. Nonetheless, even using this behaviorist–functionalist approach, there still are questions unanswered. What are the characteristics that represent the performance of this type of memory? What are the measurement criteria? How can the performance evaluation be carried out? These critical questions must be properly formulated and discussed.

2. 2 Main Features Affecting the Performance of Associative Memory

The key characteristics of the associative memory that affect the performance and need to be analyzed are the distributed information storage and collective computation. These characteristics shared by nonlinear activities and parallel process enable associative memories to solve problems that can hardly be acted on by conventional computers. However, one of the consequences of such a significant change in the memory mechanism is that the foregoing performance analyses and measure strategies used in digital computers are no longer appropriate to be applied to associative memories. As a result, developing an

investigation strategy dedicated to associative memories is one of the demands in this study. The theme of this section is to explore main features which are conceived as reflecting the performance of an associative memory. These features are characterized in a mathematical way so that corresponding performance measure can be readily undertaken. The relationships between these features are also addressed.

The following properties are considered in the evaluation of associative memory performance:

- Information capacity
- Error correction capability
- The effect of input and output pattern dimensions on accretive recall
- Spurious and oscillatory states
- System complexity in terms of architecture, encoding and recall.

2. 2 .1 Information capacity

In general, the information in an associative memory can be expressed in the form of bits or of vectors which are called patterns. But before giving a rigid definition of information capacity, two distinct but related associative recalls need to be clarified. The first is *accretive recall*:

accretive recall: Let $\{\mathbf{x}^{(s)}, \mathbf{y}^{(s)}\}$, s=1, 2, ...p, be arbitrary p pairs of associations stored in the memory, where $\mathbf{x}^{(s)} = (x_1^{(s)}, x_2^{(s)}, \ldots, x_n^{(s)})^T$, $\mathbf{y}^{(s)} = (y_1^{(s)}, y_2^{(s)}, \ldots, y_m^{(s)})^T$, and $T(.)$ be a nonlinear transformation. For a noise component, ϵ , in $\mathbf{x}^{(s)}$, a memory is said to be performing an *accretive recall* if $\mathbf{y}^{(s)}$ satisfies

$$\mathbf{y}^{(s)} = T\ (\ \mathbf{x}^{(s)} + \epsilon\ ), \qquad \forall s,\quad s=1, 2, ..., p. \tag{2.1}$$

This definition implies that the accretive recall strictly requires the stored pattern be retrieved perfectly. This type of recall is schematically depicted in Fig. 2.1 of case I. Note that the Fig. 2.1 is very idealized and, in particular, the attraction regions (shaded areas in X space) may not be circular.

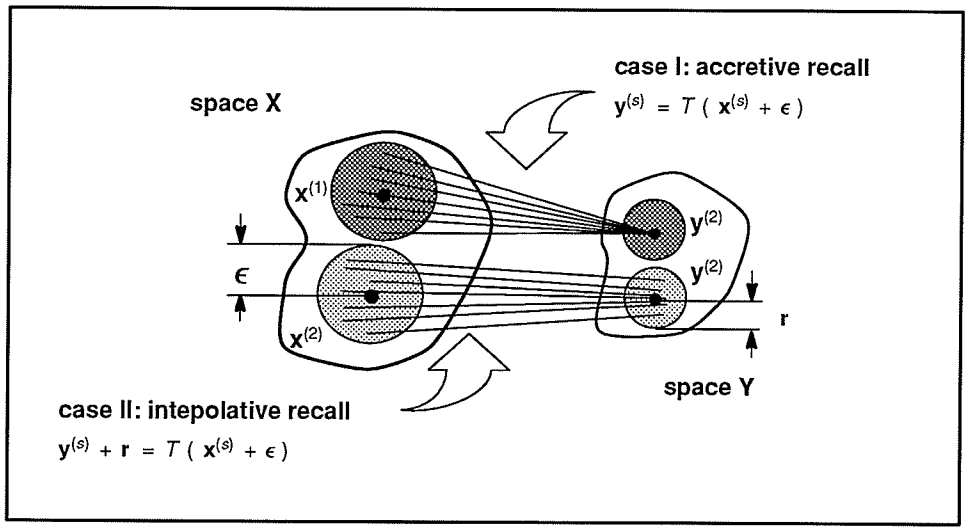

Fig. 2.1 Two dimensional representation of accretive and interpolative recall.

Another type of recall which seems to have been surprisingly neglected is the *interpolative recall.* It is worthwhile to make clear that the function of associative memory is by no means just a basic pattern association. The meaningful associations generated in the interpolative recall makes associative memory more powerful in capturing the implicitly defined relational structures. The definition of the interpolative recall is given in the following statements:

interpolative recall: Let $\mathbf{x}^{(s)}$, $\mathbf{y}^{(s)}$, s, ϵ , and $T(.)$ be defined as the same as those in the above, and r be determined according to the nearest neighbor rule [DuHa1973]. For a noise component, ϵ , in $\mathbf{x}^{(s)}$, a memory is said to be performing an *interpolative recall* if $\mathbf{y}^{(s)}$ and δ satisfies

$$\mathbf{y}^{(s)} + \boldsymbol{\delta} = T\ (\ \mathbf{x}^{(s)} + \epsilon\ )\ , \qquad \forall s,\ s=1, 2, ..., p \tag{2.2}$$

and $$0 < \| \boldsymbol{\delta} \| \leq r$$

where ||.|| denotes some proper distance measure (see case II in figure 2.1).

Eqn. (2.1) indicates that when given a stimulus, $\mathbf{x}^{(s)} + \epsilon$, the memory responds with an output which is exactly the same as the stored pattern, $\mathbf{y}^{(s)}$, associated with the input, $\mathbf{x}^{(s)}$. For this type of recall, the information capacity is defined as:

information capacity (1): The information capacity is the maximum number of patterns/associations that can be stored in the memory under the condition that when evoked by *noiseless* input, ($\epsilon = 0$), the memory can always perform accretive recall. Mathematically speaking, C_{ap} is said to be a capacity of a memory if

$$C_{ap} = \max(p) \tag{2.3}$$

such that $\mathbf{y}^{(s)} = T\ (\mathbf{x}^{(s)})$, s=1, 2, ..., p, holds.

The other way to define information capacity is on the interpolative recall basis. This definition is mainly to characterize how densely the patterns/associations can be packed in the memory under the condition that most features of the stored patterns can still be successfully retrieved.

information capacity (2): Information capacity is the maximum number of patterns/associations that can be stored in the memory under the condition that the memory is free of performing either accretive recall or interpolative recall when the memory is evoked by *noiseless* input. Mathematically speaking, C_{ap} is said to be a capacity of a memory if

$$C_{ap} = \max(p) \tag{2.4}$$

such that $\mathbf{y}^{(s)} + \boldsymbol{\delta} = T(\mathbf{x}^{(s)})$, s=1, 2, ..., p, holds. Here $\boldsymbol{\delta}$ must satisfy $0 < \| \boldsymbol{\delta} \| \leq r$. It is important to point out that, in the measure of information capacity, only specifying the number of patterns p is meaningless unless the mutual relationship between $\{\mathbf{x}^{(s_1)}, \mathbf{y}^{(s_1)}\}$ and $\{\mathbf{x}^{(s_2)}, \mathbf{y}^{(s_2)}\}$, $s_1 \neq s_2$, has been taken into account. This is because information capacity for distributed memory is usually environment dependent. The selectivity is significantly affected by the degree of mutual coupling among stored patterns. In order to get rid of such an influence it may be better to characterize this essential feature by using a probabilistic analysis approach, i.e., defining information capacity as the probability of memory being able to retrieve stored patterns perfectly (accretive recall) or imperfectly where most original features are preserved (interpolative recall).

In light of definitions of the information capacity given here, it is necessary to summarize this important property: (i) The capacity for associative memory is characterized on the probabilistic basis. This probability can be estimated using computer simulation. (ii) The information capacity is a function of p, the number of patterns stored in the memory. Thus, the curve, the probability of associative recall vs. the number of stored patterns p, directly

manifests the memory capacity.

2. 2 .2 Error correction capability

The most appealing feature in associative memory is its ability to suppress noise or to recover most information correctly even when the memory is stimulated by an incomplete or noise corrupted input pattern. This property reflects memory's "thinking" capability. The ability to correct noise is achieved by means of various techniques depending on the type of architecture, encoding and retrieving algorithm used. Orthogonal projection [Koho1972] [Koho1984], lateral inhibition [Lipp1987], [RuMc1986], steep descent iteration [HiAn1989] and bidirectional feedback search [Kosk1987], [Hass1989] are all well–known and widely utilized techniques in building associative memories. The quality of associative recall can be improved either by the activity of competition among neurons during the recall or by performing signal feedback to force a memory to dynamically evolve until it reaches a stable state. The capability of correcting noise is usually characterized by whether the memory is able to give a high quality response despite variations, distortions and omissions in the input pattern.

The distinction between information capacity and error correction capability lie in that the information capacity is to exhibit how densely the information can be stored into a memory, whereas the error correction is to manifests the associative memory being able to reconstruct stored patterns/associations when the memory is stimulated by degraded or partially absent input patterns. Furthermore, the capacity is measured under the condition that all input patterns are purely the training patterns themselves, whereas the error

correction capability is characterized by the relationship between the quality of associative recall and the noise level in input patterns.

One thing that needs to be emphasized is that the criterion for judging whether memory performs successful recall (either accretive or interpolative recall) is based on the nearest neighbor rule [DuHa1973]. The nearest neighbor rule simply classifies a pattern $\mathbf{x}$ according to the nearest point in the training set. Here the "nearest" is defined on the norm based computation; unless otherwise specified, it is the Hamming distance measure. For two patterns $\mathbf{x}^{(s1)} = (x_1^{(s1)}, x_2^{(s1)}, \ldots, x_n^{(s1)})$ and $\mathbf{x}^{(s2)} = (x_1^{(s2)}, x_2^{(s2)}, \ldots, x_n^{(s2)})$, the Hamming distance, $D_H(\mathbf{x}^{(s1)}, \mathbf{x}^{(s2)})$, is calculated according to

$$D_H(\mathbf{x}^{(s1)}, \mathbf{x}^{(s2)}) = \sum_{i}^{n} | x_i^{(s1)} - x_i^{(s2)} | . \tag{2.5}$$

The nearest neighbor rule applied to judging whether or not an associative memory performs successful recall is formulated by

$$\begin{aligned} &\mathbf{y}^{(recall)} = \mathbf{T}(\mathbf{x}^{(k)} + \epsilon) \textit{ is said to be recalled successfully} \\ &\text{if } \mathbf{y}^{(recall)} \textit{ satisfies :} \\ &(1)\ \exists \mathbf{y}^{(k)} \textit{ such that } \mathbf{y}^{(k)} \in \{\mathbf{y}^{(s)}\}, \quad s = 1, 2, \ldots, p, \textit{ and} \\ &(2)\ \| \mathbf{y}^{(recall)} - \mathbf{y}^{(k)} \| = \min_{s} \| \mathbf{y}^{(recall)} - \mathbf{y}^{(s)} \| . \end{aligned} \tag{2.6}$$

Eqn. (2.6) indicates that, for a memory having stored p associations $\{\mathbf{x}^{(s)}, \mathbf{y}^{(s)}\}$, s=1, 2, ..., p, when given a stimulus, $\mathbf{x}^{(init)}=\mathbf{x}^{(k)}+\epsilon$, the recalled pattern, $\mathbf{y}^{(recall)}$, is classified to $\mathbf{y}^{(k)}$ if $\mathbf{y}^{(recall)}$ is the nearest neighbor of $\mathbf{y}^{(k)}$, $k \in \{1, 2, \ldots, p\}$. Evidently, if $\| \mathbf{y}^{(recall)} - \mathbf{y}^{(k)} \| = 0$, it is an accretive

recall because $\mathbf{y}^{(k)}$ is one of the stored patterns. If $\| \mathbf{y}^{(recall)} - \mathbf{y}^{(k)} \| \neq 0$, it must be an interpolative recall.

2. 2 .3 The effect of input and output pattern dimensions on accretive recall.

Due to the distributive manner of information storage, the memory performance may be sensitive to the input and output pattern dimensions. This is because it is only the input pattern that is responsible for furnishing information to evoke the memory. Generally speaking, the more redundant the information (the higher the dimension in the input pattern) with which the input pattern can be represented, the higher the probability the memory performs correct recall. However, the degree of such an effect varies depending on different models, the number of associations stored in the memory, and the noise level in input patterns. For this reason, any measurement of this effect of input and output pattern dimensions on accretive recall is not meaningful unless all required conditions are specified. In order to quantitatively measure such an effect, the following definition is proposed:

Let the effect of input and output pattern dimensions on accretive recall between two memory models A and B be denoted by Z_{AB} and n and m be the dimension of the input and output patterns, respectively. If the model A is constructed with $m \neq n$, while the model B is constructed with $m = n$, then Z_{AB} can be defined as:

$$Z_{AB} = C\ S(AB) \left(\frac{1}{l-1} \sum_{k=1}^{l} \left(A_{pe}^{pk} - B_{pe}^{pk} \right)^2 \right)^{1/2} \tag{2.7}$$

where $l \geq 2$ and C is a constant (in the investigation, C=10 is used). The function $S_{(AB)}$ in Eqn. (2.7) is defined as

$$S(AB) = sgn\left(\sum_{k=1}^{l} (A_{p_e}^{p_k} - B_{p_e}^{p_k}) \right) \tag{2.8}$$

and $A_{p_e}^{p_k}$, k=2, 3, ..., l, is the probability of accretive recall measured under the following conditions: (i) memory structure: $m \neq n$, (ii) input noise level: p_e, and (iii) memory load: $p_1, p_2, ..., p_l$. Each p_k, k=2, 3, ..., l, stands for the number of training patterns stored in the memory and $p_i < p_j$ if $i < j$. $B_{p_e}^{p_k}$ is also the probability of accretive recall measured under the same conditions as $A_{p_e}^{p_k}$ except for the memory structure: $m = n$. The Eqn. (2.7) actually measures the performance differences (in the accretive recall aspect) between the model A and model B. This difference measure is analogous to calculating the sample standard deviation. In Eqn. (2.7) the number of observations is l, and the performance of model B with the same dimension in its input and output (m=n) is treated as a sample mean (used as a reference). The function, $S_{(AB)}$, used in Eqn. (2.7) is intended to show whether such an effect improves or deteriorates the quality of accretive recall.

2. 2 .4 Spurious and oscillatory states

False states can be divided into two categories: the spurious states and the oscillatory states. Spurious memories are those stable states that do not belong to stored training patterns. These states are generated during the encoding process. Thus, the energy surface for some types of memory is not only determined by training patterns, but also shaped by

spurious states. These states with relatively lower energy potential affect memory performances by attracting input patterns or intermediate states into their local minima. The property of this phenomenon in the process of associative recall is characterized by

$$\mathbf{y}_{(t+1)}^{(recall)} = \mathbf{y}_{(t)}^{(recall)}, \qquad \forall t,\ t > t_0$$

and $$\mathbf{y}_{(\infty)}^{(recall)} \neq \mathbf{y}^{(s)}, \qquad \forall s,\ s=1, 2, ..., p \tag{2.9}$$

here t is the iteration number. It is worth noting that so far, there has been no method to directly control the location and to minimize the number of spurious memories. Choosing a proper encoding algorithm to solve a particular problem may be an alternative way to reduce these unexpected stable states.

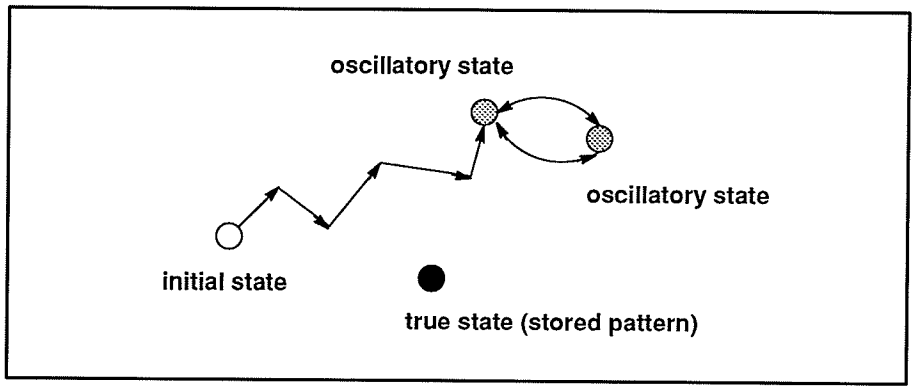

Fig. 2.2 Illustration of memory converging to oscillatory state.

That the associative memory converges to oscillatory memory is another phenomenon which needs to be taken into account. This phenomenon only takes place in those memory models in which iterative or dynamic bidirectional recall [Hass1989], [Kosk1988] is used. For both a synchronous and an asynchronous adaptive recall, a memory may take several iterations before reaching a stable state. Therefore, there exists a possibility that memory

converges neither to a true state nor to a spurious state but to an oscillatory state. Fig. 2.2 shows one possible convergence case that may occur in dynamic associative memories.

The activity of oscillation in a dynamic bidirectional associative memory can be formulated by

$$\mathbf{x}_{(t_0)} \rightarrow \mathbf{y}_{(t_0)} \rightarrow \mathbf{x}_{(t_1)} \rightarrow \mathbf{y}_{(t_1)} \rightarrow \ldots \rightarrow \mathbf{x}_{(t_k)} \rightarrow \mathbf{y}_{(k_k)} \rightarrow \mathbf{x}_{(t_0)} \tag{2.10}$$

here $k = t_k - t_0$ denotes the oscillatory period.

2. 2 .5 System complexity

Specifically, the study of system complexity involves analyzing spatial complexity, encoding and recalling temporal complexity. Spatial complexity usually refers to the physical structure of a memory system. Primary factors that determine spatial complexity are the physical structure, i.e., the number of layers in the memory, and the number of neurons in each layers. The neurons in the memory may be fully connected or partially connected. These alternative choices in terms of the memory structure depend on the application at hand.

The second aspect associated with the system complexity is the speed of encoding. The ability of keeping pace with other machines and completing learning within a limited time period determines whether the memory can be applied to solving real–time problems. The encoding process deals with teaching an associative memory how to behave or react when the memory is stimulated. It is realized by forming or adjusting a numerical version of synaptic weights in a software implementation or an electronic version in a VLSI hardware implementation. Thus, the amount of time for a memory to organize its own internal

structure depends largely on the complexity of the encoding algorithm employed. Some encoding algorithms simply look for similarity or correlation in a set of training patterns [Hopf1982]. Others adopt trial–and–error, a step by step error correcting approach to adjust weights. The former algorithm requires extremely low computational time. The latter algorithm allows memory for achieving higher performance but at the cost of substantially longer execution time [RuMc1986].

The third issue with system complexity that needs to be discussed is the speed of recall. This is concerned with the processing time used during the retrieval period. For a feedforward memory, the computational time required in associative recall is simply determined by the network architecture, the number of processing elements and the connection fashion (full or partial). For a dynamic associative memory, however, the number of iterations must be taken into account.

It is known that directly recording the execution time to measure the temporal complexity may be inevitably affected by the computer characteristics. If two algorithms are compared first on one machine and then another, the comparisons may lead to different conclusions. To avoid this machine dependent measurement, it is preferable to use the mathematical analysis.

All properties discussed above are critical to characterize the performance of associative memory. However, only four of them: information capacity, error correction capability, the probability of memory getting stuck at spurious states and the probability of memory converging to oscillatory states have been thoroughly investigated in this study. These items are chosen because they are exclusive from those of localized memories as well as because they have seldom been considered sufficiently.

2. 3 Procedure for Performance Evaluation

The above descriptions in some sense still remain at the conceptual standpoint. To carry out empirical performance evaluation, a specific procedure need to be developed. In this section, an attempt is made to meet this requirement by providing every step in detail.

2. 3 .1 Testing pattern synthesis

The fundamentally active entities in associative memory are "state vectors". Elements of vectors are generally considered to be the magnitude of activity in a particular neuron. At the current stage of technology, most existing neural network models of associative memory employ a hardlimiter or a unit step activation function to realize noise suppression as well as to help systems to reach stable states in adaptive recall. For some encoding algorithms [Koho1972], [RuMc1986], real value mapping is achievable but the capability of error correction is very limited. Instead of correcting error, generalization or interpolation is the alternative property inherent in these systems. For associative memories, these properties are deemed a drawback (in terms of accretive recall). For this reason, at the current stage, one has to restrict his attention to a binary or a bipolar mode pattern. In the following discussion, information entities are all assumed as binary or bipolar patterns unless otherwise specified.

There are two groups of testing patterns needed to be generated: the training patterns to be stored in the memory and the testing patterns used to stimulate memories. In this subsection only the first group of testing patterns is addressed. Generating the second group of testing patterns will be discussed in the subsection 2.3.2.

Obtaining the characteristics of information capacity by means of simulative testing usually needs a large number of training sets. The maximum number of training patterns/associations, p_{max} , required to estimate the capacity of associative memory depends on the memory model being tested. One approach to the reduction of the computation burden associated with the probabilistic method is to monitor the memory performance while testing (instead of using fixed p_{max}). Such a heuristic testing approach usually starts from a small number p, and then p is increased by a fixed step. The increment may be operated by the program itself. The testing terminates if the memory performance drops down to a predefined criterion.

The other aspect that needs to be considered is how to generate these patterns. In most computer simulation, such as simulating a communication system, random variables with desired size and distribution are usually specified [LaKe1982]. Instead of purely selecting training patterns randomly, it is sometimes helpful to add a constraint primarily for the purpose of reducing testing time and achieving high reliability in the test results. To achieve this goal, a method called *minimum distance constraint* (*MD*) is developed. This approach can be simply described as generating a set of n dimensional training patterns under the following conditions:

- Training patterns are uniformly distributed in n dimensional hypersphere (for a discrete pattern, this means that the chance for every pattern to be generated on any corner of n dimensional hypercube is equally likely).

- Distance between any two training patterns satisfies: $d\ (\mathbf{x}^{(s_1)}, \mathbf{x}^{(s_2)}) \geq MD$ in a memory input space, and $d\ (\mathbf{y}^{(s_1)}, \mathbf{y}^{(s_2)}) \geq MD$ in a memory output space. Here $d\ (\mathbf{x}^{(s_1)}, \mathbf{x}^{(s_2)})$ and $d\ (\mathbf{y}^{(s_1)}, \mathbf{y}^{(s_2)})$ are the norm based distance measurements.

The necessity and validity of employing the minimum distance approach are given in the following explanations. First of all, think of an autoassociative memory to perform self–reconstruction. The memory is first loaded with two patterns $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$. If $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ are far from each other in terms of distance measure, the memory can readily retrieve any one of the stored patterns with high probability. This is because the corresponding energy wells for $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ in n dimensional vector space must be sufficiently distant from each other. Moreover, the high energy potential between these two energy wells ideally separates the entire energy space into two subregions (see Fig. 2.3).

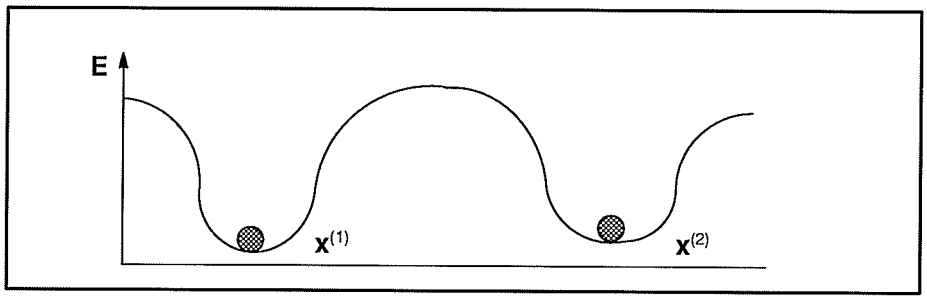

Fig. 2.3 Representation of two uncorrelated training patterns in a two dimensional energy plane.

However, if training patterns are generated purely at random, there is a possibility that two patterns are correlated or near–correlated, namely, two patterns are identical or only a few components are different. A two dimensional energy curve corresponding to a memory storing two such patterns is shown in Figure 2.4.

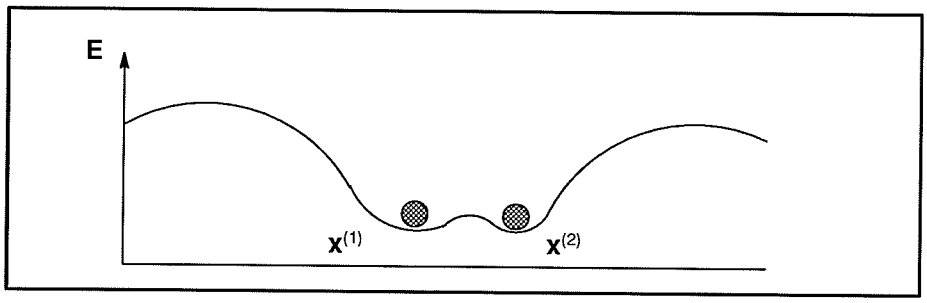

Fig. 2.4 Representation of two near correlated training patterns in a two dimensional energy plane.

It is seen that energy wells associated with $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ are close to each other. They may even be mixed up in the worst case. It is very difficult for a memory to retrieve such highly correlated patterns. Part of the reason for this is that neither synchronous nor asynchronous recall can assure that every state movement can be exactly in the correct direction, i.e., to the memorized stable state in terms of nearest neighbor from the starting state. Therefore, the memory may eventually converge to $\mathbf{x}^{(2)}$ instead of $\mathbf{x}^{(1)}$, even though the initial state is closer to $\mathbf{x}^{(1)}$. Fig. 2.5 is the two dimensional illustration of this situation. A typical example

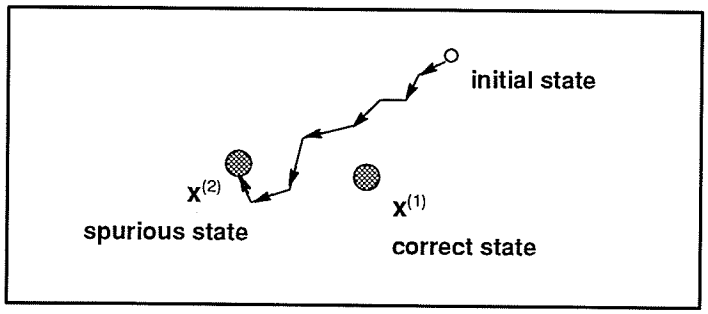

Fig. 2.5 Illustration of memory converging to a spurious state.

of this problem can be recall either of two patterns having stored in a memory. One is the lowercase character "l", the other is the numerical number "1". One has to admit that even for humans, this problem is not an easy task. One can not give a definite answer which of the patterns have been recalled unless more information, such as background or context, has been collected. However, obtaining additional information, such as context in our example, can only be achieved by a dynamically sequential process, which is usually referred to as

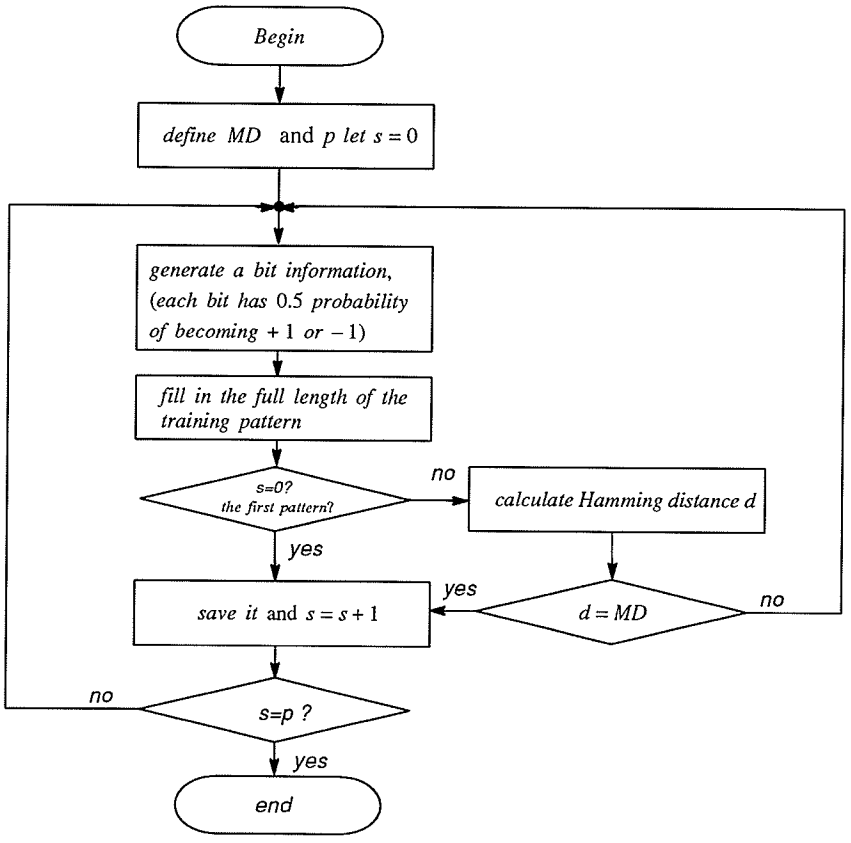

Fig. 2.6 Procedure of generating training patterns under the minimum distance constraint.

temporal recall. This leads to another type of associative memory, called *Temporal Associative Memory* [HiAn1989] [Koho1984]. Since discussing this type of a memory falls outside of this study, it will not be included in this thesis. The memory models addressed here are restricted to those working on the similarity or correlation measurement basis.

It should be noted that using the minimum distance approach to generate training patterns does not affect the uniform distribution form as long as the first pattern is selected randomly with a probability equal to $1/2^n$. Procedure for generating discrete training patterns with binary values {0, 1} or {–1, 1} is presented in Figure 2.6.

2. 3 .2 Designing test strategy

Once training patterns have been loaded into the memory by applying a learning algorithm, the next step is to generate proper testing patterns which could ideally cover any possible situation that the system may encounter in a real–world. For a sufficiently small memory, i.e. $n \leq 16$, this may allow the use of an exhaustive testing approach which is guaranteed to give complete information about performance characteristics. This approach is implemented by testing all possible states in n dimensional hypercube for each stored pattern. Such a procedure used for exhaustively testing the property of error correction capability is given in Figure 2.7.

However, the exhaustive testing approach is not applicable for a large system. Computation complexity for exhaustive testing is $O(2^n p)$. It is easy to show that execution time goes higher rapidly as n becomes larger. For instance, to simulate a system with n=120

and p=10, total testing trials would be $(C_0^{120} + C_1^{120} + C_2^{120} + ... + C_{120}^{120}) \times 10 = 2^{120} \times 10 \approx 10^{17}$ which is obviously not practicable.

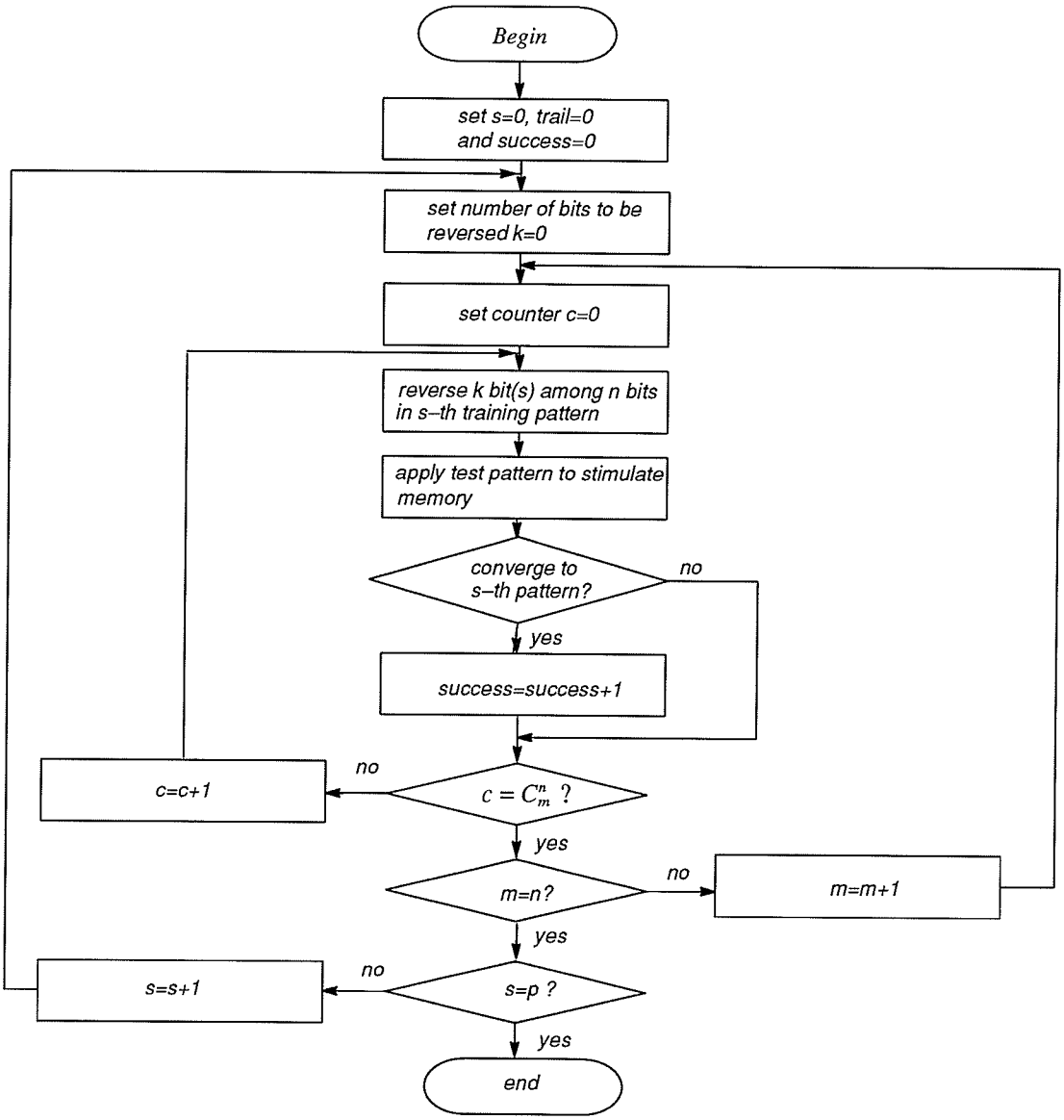

Fig 2.7 Exhaustive testing procedure for evaluating the error correction capability. s is training pattern indicator, m denotes the number of bits being reversed and c is a counter used to check whether all combinations of m bits from n components have been exhaustively reversed.

To derive an effective test strategy, define $P(e \mid \mathbf{x}^{(s)})$, s=1, 2, ..., p, a conditional error probability of system output in response to the input pattern $\mathbf{x}^{(s)}$. Because

- event $\mathbf{x}^{(i)} \cdot \mathbf{x}^{(j)}=0$, if $i \neq j$;
- $\mathbf{x}^{(1)} + \mathbf{x}^{(2)} + \mathbf{x}^{(3)} ... + \mathbf{x}^{(p)}$ = sample space

according to the theory of conditional probability (the Bayes' rule) [Triv1982], the system error probability can be defined as

$$P_r(e) = \sum_{s=1}^{p} P_r(e \mid \mathbf{x}^{(s)}) P_r(\mathbf{x}^{(s)}) \tag{2.11}$$

where $P_r(\mathbf{x}^{(s)})$ denotes the probability of training pattern $\mathbf{x}^{(s)}$, s=1, 2, ..., p, being selected during the testing. If the pattern in the training set is selected independently and each pattern is tested 2^n times (by reversing the all possible combination of k bit(s) from it), the overall testing time will be $p \times 2^n$. Since

$$P_r(\mathbf{x}^{(1)}) = P_r(\mathbf{x}^{(2)}) = \ldots = P_r(\mathbf{x}^{(p)}) = \frac{2^n}{p \times 2^n} = \frac{1}{p}, \tag{2.12}$$

Eqn. (2.11) can be simplified as

$$P_r(e) = \frac{1}{p} \sum_{s=1}^{p} P_r(e \mid \mathbf{x}^{(s)}) . \tag{2.13}$$

Actually $P_r(e \mid \mathbf{x}^{(1)}) \approx P_r(e \mid \mathbf{x}^{(2)}) \approx \ldots \approx P_r(e \mid \mathbf{x}^{(p)})$. This is because

- The distance between any pairs of $\mathbf{x}^{(s_1)}$ and $\mathbf{x}^{(s_2)}$ $(s_1 \neq s_2)$ has been set to MD ; hence, nothing affects the nature of $\mathbf{x}^{(s)}$ except in the geometrical aspect. However, memory performance influenced by this factor is very limited. Thus, it can be discounted.
- Testing procedure applied to derive $P_r(e \mid \mathbf{x}^{(s_1)})$ is identical to those being applied to $P_r(e \mid \mathbf{x}^{(s)})$, where s=1, 2, ..., p and $s \neq s_1$.

The forgoing approximation yields a simplified form of system error probability

$$P_r(e) = \frac{1}{p} \; p \; P_r(e \mid \mathbf{x}^{(s)}) \; = P_r(e \mid \mathbf{x}^{(s)}) \tag{2.14}$$

where s can be any one of $\{1, 2, ..., p\}$.

Equation (2.14) implies that system error probability can validly be obtained by testing any one of the stored patterns. This approach allows for the reduction in execution time which is equal to $(p-1)2^n$, $(p \geq 2)$. Clearly, significant computation time can be saved as p grows. However, this approach can not compete with the exhaustive testing approach in terms of coverage. The result can be no more than an estimator of true performances.

However, the complexity function $O(2^n)$ is evidently not a small cost. The significant growth of the complexity as n increases unavoidably results in inability to terminate evaluation in an acceptable period of time. The method used to generate noisy input in this work is starting from a stored pattern and then adding noise into it. For a binary pattern, adding noise can be implemented by randomly reversing bits in an input pattern according to a certain probability, i.e., chance for each bit to be reversed is equally likely with a specified probability, P_e . According to the theory of Binomial distribution [Ross1987], the characteristic of a noisy pattern can be described in a probabilistic way. The probability for

k bit(s), $1 \leq k \leq n$, being reversed in n dimensional pattern can be calculated by Eqn. (2.15) provided the probability for each bit being reversed P_e is known.

$$P_r\{x = k\} = \binom{n}{k} P_e^k \, (1 - P_e)^{n-k} \tag{2.15}$$

Throughout the testing, the probability of each bit being reversed, P_e, is set to 0.01, 0.07, 0.1, 0.2, 0.3, 0.4 and 0.5. Their corresponding probability of k bit(s) being reversed from an input pattern with n=16 are given in Table 2.1 .

Table 2.1 The Probability of k Bit(s) to Be Reversed from a Set of 16 Bits

P_e	The number of bit(s) being reversed								
	0	1	2	3	4	5	6	7	8
0.01	0.85146	0.13761	0.01043	0.00049	0.00002	0.00000	0.00000	0.00000	0.00000
0.07	0.31313	0.37711	0.21288	0.07478	0.01829	0.00330	0.00046	0.00005	0.00000
0.1	0.18530	0.32943	0.27452	0.14234	0.05140	0.01371	0.00279	0.00044	0.00006
0.2	0.02815	0.11259	0.21111	0.24629	0.20011	0.12007	0.05503	0.01966	0.00553
0.3	0.00332	0.02279	0.07325	0.14650	0.20405	0.20988	0.16490	0.10096	0.04868
0.4	0.00028	0.00301	0.01505	0.04681	0.10142	0.16227	0.19833	0.18889	0.14167
0.5	0.00015	0.00024	0.00183	0.00855	0.02777	0.06666	0.12219	0.17456	0.19638

The next parameter that needs to be determined is the number of trials required for each P_e in order to ensure reliability. Throughout the testing, the number of trials, C_P, for particular defined noise probability, P_e , is calculated according to [Jeru1984], [HeNo1988]

$$C_P = \frac{70}{P_r(\ err\ |P_e\ )} \tag{2.16}$$

where P_r (*err* | *Pe*) is the error probability of associative recall under the condition that the probability for each bit being reversed is P_e.

The last parameter to be specified is the number of trials for each case, i.e., for a memory loaded with p stored patterns/associations, how many groups of training sets (with p patterns/associations each) are needed so that reliable information about memory characteristics can be obtained. This parameter determines the terminating time for the testing. A usual approach to select this parameter is based on the fixed–sample–size procedure [LaKe1982]. This procedure is described as in the following: a simulation run of an arbitrary fixed length is performed, and then one of the sample sizes that satisfies the desired confidence interval is finally selected [LaKe1982]. In this work, the above fixed–sample–size procedure is adopted to determine the terminating time. Since this parameter is task dependent, the detailed discussion about this issue will be given in section 3.2 in chapter 3.

2. 3 .3 Procedure used to test the performance of associative memory

This subsection presents complete procedure used for testing the performance of associative memories. The results of executing the procedure are the performance statistics including the probability of memory successfully performing accretive and/or interpolative recall, the probability of memory converging to spurious states, and the probability of memory converging to oscillatory states. These results reflect the overall performances of

the memory being tested. Some of the performance characteristics, such as information capacity and error correction capability, may be considered as the extreme–performance if the maximum value of the minimum distance is used. The rule of classifying a memory response into a proper feature category is based on the definitions given in section 2.2.1–2.2.4. The summarized classification strategy is described in the following statements:

(i) *Accretive recall* stands for the stored pattern/association that is completely recalled, namely,

$$\mathbf{x}^{(k)} + \epsilon \rightarrow \mathbf{y}^{(recall)} = \mathbf{y}^{(k)} \tag{2.17}$$

where $(\mathbf{x}^{(k)}, \mathbf{y}^{(k)}), (\mathbf{x}^{(k)}, \mathbf{y}^{(k)}) \in \{\mathbf{x}^{(s)}, \mathbf{y}^{(s)}\}$, s=1, 2, ..., p, is the training pair stored in the memory and ϵ refers to the noise component contained in input pattern. The term $\mathbf{x}^{(s)} + \epsilon$ is the testing pattern used to stimulate the memory. The recalled pattern at the memory output is denoted by $\mathbf{y}^{(recall)}$.

(ii) *Interpolative recall* is defined as a mapping such that the desired pattern, $\mathbf{y}^{(k)}$, is the nearest neighbor of a recalled pattern $\mathbf{y}^{(recall)}$, i.e.,

$$\begin{aligned} &\mathbf{x}^{(k)} + \epsilon \rightarrow \mathbf{y}^{(recall)} \\ \text{and} \quad &\| \mathbf{y}^{(recall)} - \mathbf{y}^{(k)} \| = \min_{s} \| \mathbf{y}^{(recall)} - \mathbf{y}^{(s)} \| . \end{aligned} \tag{2.18}$$

(iii) *Spurious states* are the stable states but they are exclusively from the training patterns/associations. Whenever Eqn. (2.19) holds, the spurious states should be claimed.

$$\mathbf{x}^{(k)} + \epsilon \rightarrow \mathbf{y}^{(recall)} \neq \mathbf{y}^{(s)} , \qquad \forall s \; s = 1, 2, \ldots, p . \tag{2.19}$$

Spurious states can further be categorized into two different categories. Spurious states in one of the categories constitute successful mapping (in interpolative recall case only. Refer section 2.2.1 for more details). Spurious states in the other category cause false mapping. A false spurious state is detected if the recalled pattern, $\mathbf{y}^{(recall)}$, satisfies the following conditions:

$$\begin{aligned} & \mathbf{x}^{(k)} + \epsilon \rightarrow \mathbf{y}^{(recall)} \neq \mathbf{y}^{(k)}, \qquad \forall s,\ s = 1, 2, \ldots, p \\ \text{and} \quad & \| \mathbf{y}^{(recall)} - \mathbf{y}^{(k)} \| \neq \min_{s} \| \mathbf{y}^{(recall)} - \mathbf{y}^{(s)} \| . \end{aligned} \tag{2.20}$$

(iv) *Oscillatory states* can be detected whenever a memory fails to converge to a stable state. This is done by checking if $\mathbf{y}^{(recall)}$ satisfies

$$\begin{aligned} & \mathbf{y}_{(t+1)}^{(recall)} \neq \mathbf{y}_{(t)}^{(recall)}, \qquad \forall t,\quad t > 0 \\ \text{and} \quad & \mathbf{y}_{(t+t_k)}^{(recall)} = \mathbf{y}_{(t)}^{(recall)}, \qquad t_k \in N \text{ and } t_k \geq 2 . \end{aligned} \tag{2.21}$$

If Eqn. (2.21) holds, it means the memory becomes oscillation.

The procedure designed to estimate the true performance characteristics of associative memories is presented below. It should be pointed out that the maximum number of training patterns/associations, $p_{\max}$, used in the testing is determined by the performance of the memory being tested. The performance criterion, $P_{criterion}$, used to terminate the testing is specified before the testing begin. The number of training patterns/associations starts from a small number, and then it keeps increasing by a fixed step. The testing terminates if the memory performance drops down to $P_{criterion}$.

Special Notations for the Procedure of Testing the Performance of Associative Memory

.Symbol	Denotes
. *acc*	Stored pattern is completely retrieved (accretive recall).
. c_1	Counter for T_p .
. c_2	Counter for C_P .
. *fsp*	False spurious states.
. C_P	Number of trials for each P_e.
. T_p	The number of trials under the condition that memory is loaded with *p* pairs of associations, $\{\mathbf{x}^{(s)}, \mathbf{y}^{(s)}\}$, *s*=1, 2, ..., *p*).
. *int*	Stored pattern is partially retrieved (interpolative recall).
. *p*	The umber of training patterns attempted to be stored in the memory.
. $p_{\min}$	The minimum value of *p*.
. $P_{criterion}$	The performance criterion for terminating testing.
. P_e	The probability of each bit (in input pattern) being reversed. (P_e is set to 0, 0.01, 0.07, 0.1, 0.2, 0.3, 0.4 and 0.5).
. *os*	Oscillatory states.
. *sp*	Spurious states.
. *trials*	Total trials.
. ϵ	Noise component in an input pattern.

Procedure for Testing the Performances of Associative Memory

Step (1) Set up T_p , and value of *MD* for each *p*.

Step (2) Specify P_e .

Step (3) Set up the performance criterion for terminating testing, $P_{criterion}$ (the probability of associative recall)

Step (4) Set $p = p_{\min}$, the number of patterns to be stored in the memory.

Step (5) While the probability of associative recall (either accretive or interpolative) is higher than criterion $P_{criterion}$ execute following loop:

Step (5.1) Estimate C_P according to equation (2.16).

Step (5.2) Set counter c_1=0.

Step (5.3) Set *trials* = 0.

Step (5.4) Set *acc* = 0, *int* = 0, *sp*=0, *fsp*=0, *os*=0.

Step (5.5) Generate *p* pairs training associations $\{\mathbf{x}^{(s)}, \mathbf{y}^{(s)}\}$ *s=1, 2, ..., p* based upon minimum distance rule described in Section 2.3.1.

Step (5.6) Store information into the memory using encoding algorithm.

Step (5.7) Randomly select one of the associations, $(\mathbf{x}^{(k)}, \mathbf{y}^{(k)})$, from *p* training pairs, i.e., $(\mathbf{x}^{(k)}, \mathbf{y}^{(k)}) \in \{\mathbf{x}^{(s)}, \mathbf{y}^{(s)}\}$, *s*=1, 2, ..., *p*. The chance for each pair being selected is equally likely.

Step (5.8) Set counter c_2 = 0.

Step (5.9) Add noise, ϵ , into input pattern, $\mathbf{x}^{(k)}$, by means of randomly reversing each bit in $\mathbf{x}^{(k)}$ according to defined probability P_e .

Step (5.10) Apply testing pattern, $\mathbf{x}^{(k)} + \epsilon$, to the memory input.

Step (5.11) Recall.

Step (5.12) Check intermediate state in each iteration. If satisfy Eqn. (2.21) increase *os*.

Step (5.13) Collect response at the output. if $\mathbf{y}^{(recall)} = \mathbf{y}^{(k)}$ increase *acc*, otherwise increase *sp*.

Step (5.14) If $\| \mathbf{y}^{(recall)} - \mathbf{y}^{(k)} \| = \min_{s} \| \mathbf{y}^{(recall)} - \mathbf{y}^{(s)} \|$ increase *int*, otherwise increase *fsp*.

Step (5.15) Increase *trials*.

Step (5.16) If $c_2 < C_P$ increase c_2, and then go to step (5.9).

Step (5.17) If $c_1 \leq T_p$ go to step (5.3), otherwise, next step.

Step (5.18) Calculate performance statistics under the condition that the probability of each bit being reversed is P_e : (i) the probability of accretive recall: *acc / trials*, (ii) the probability interrogative recall*: int / trials*, (iii) the probability of memory converging to spurious states: *sp /trials*, (iv) the probability of memory converging to false spurious states: f*sp / trials*. (v) the probability of memory converging to oscillatory states: *os / trails*.

Step (5.19) increase *p* if the probability of associative recall (either accretive or interpolative) is higher than performance criterion $P_{criterion}$.

Step (5.20) Until the probability of associative recall is lower than the performance criterion $P_{criterion}$.

Step (6) Choose another P_e and then restart procedure from step (4) until all defined P_e has been tested.

Step (7) End of procedure.

Note: This procedure test the information capacity, spurious states and oscillatory states. Because the error correction capability is concerned with the relationship between the quality of associative recall and input noise, this feature can be shown by plotting the probability of associative recall against the noise level (the probability of each bit being

reversed in input patterns) used in the testing. All data required will be available after executing the above procedure.

2. 4 References

[DuHa1973] Duda, R. O. and Hart, P. (1973), "Pattern Classification and Scene Analysis", John Wiley & Sons.

[HaYo1989] Hassoun, M. H. and Youssef, A. M. (1989), "High performance recording algorithm for Hopfield model", Optical Engineering, Vol. 27, No.1, pp 46–54.

[Hass1989] Hassoun, M. H. (1989), "Dynamic heteroassociative memories", Neural Networks, Vol. 2, pp. 275–287.

[HeNo1988] Herro M. A. and Nowack J. M. (1988), "Simulated Viterbi decoding using importance sampling", IEE Proceedings, Vol. 135, Pt. F, No. 2, pp. 133–142.

[HiAn1989] Hinton, G. E. and Anderson, J. A. (Eds.) (1989), "Parallel Models of Associative Memory", Updated Edition, Hillsdale, N. J. : Lawrence Erlbaum Associates.

[Hopf1982] Hopfield, J. J. (1982), "Neural networks and physical systems with emergent collective computational abilities", Proceedings of the National Academy of Science USA, Vol. 79, pp. 2554–2558.

[Hopf1984] Hopfield, J. J. (1984), "Neurons with graded response have collective computational properties like those of two–state neurons", Proceeding of the National Academy of Sciences, U.S.A., Vol. 81, pp. 3088–3092.

[Jeru1984] Jeruchim, M. C. (1984), "Techniques for estimating the bit error rate in the simulation of digital communication systems", IEEE Journal Selected Areas in Communication, SAC–2, pp. 153–170.

[KlPC1988] Klassen, M. Y., Pao, H. and Chen, V. (1988), "Characteristics of the functional link net: A higher order delta rule net", Proceedings of IEEE International Conference on Neural Networks, Santiago, CA: SOS Print.

[Koho1972] Kohonen, T. (1972), "Correlation matrix memories", IEEE Transactions on Computers, Vol. 21, pp. 353–359.

[Koho1977] Kohonen, T. (1977), "Associative Memory, a System–Theroetical Approach", Berlin: Springer–Verlag.

[Koho1984] Kohonen, T. (1984), "Self–organization and Associative Memory", Berlin: Springer–Verlag.

[Kosk1988] Kosko, B. (1988), "Bidirectional associative memories", IEEE Transactions on System, Man and Cybernetics, Vol. 18, No. 1, pp. 49–60.

[Kron1987] Kronsjo, L. (1987), "Algorithm: the Complexity and Efficiency", Second Edition. John Wiley & Sons.

[LaKe1982] Law, A. M. and Kelton, W. D. (1982), "Simulation Modeling and Analysis", McGraw–Hill Book Company.

[Lipp1987] Lippmann, R . P. (1987), "An introduction to computing with neural nets", IEEE ASSP Magazine, April, pp. 4–22.

[Ross1987] Ross, S. M. (1987), "Introduction to Probability and Statistics for Engineers and Scientists", John Wiley & Sons.

[RuMc1986] Rumelhart, D. E., McCelland, J. L and the PDP Research Group, (1986), "Parallel Distribution Processing: Exploration in the Microstructure of Cognition, Vol. 1, Foundations", The MIT Press Cambridge, Massachusetts.

[Triv1982] Trivedi, K. S. (1982), "Probability and Statistics with Reliability, Queuing and Computer Science Applications", Printice–Hall.

CHAPTER 3

TESTED MODELS AND EXPERIMENTAL RESULTS

3. 1. Introduction

This chapter describes a series of experiments investigating major properties of information capacity, error correction capability, the effect of input and output pattern dimensions on accretive recall, and the probability of memory converging to false states. Altogether, four different models were involved in the experiments. They were the Hopfield networks [Hopf1982], [Hopf1984], bidirectional associative memories (BAMs) [Kosk1987], [Kosk1988], dynamic associative memories implemented by the Ho–Kashyap's algorithm (HK Models) [Hass1989], and backpropagation networks (BP networks) [RuMc1986]. It is necessary to point out that choosing these fundamental memory models does not mean that testing procedure derived in chapter 2 is only suitable for these systems. They are networks typically designed for associative memories except for the backpropagation network which has more applications in other areas.

Although several performance investigations were carried out previously, most of them merely involved one or two benchmark data set(s) to demonstrate the robustness of newly developed memory systems [Lipp1987], [Kosk1987], [Kosk1988]. Since these investigations were undertaken independently by different individuals, it was very difficult

to judge which one was superior to others based on these fragmented results. Valid comparison can only be carried out under the same environment and by using the same evaluation scheme. To establish the superior performance characteristics of the HK model, M. H. Hasson performed a number of simulations [Hass1989]. His work successfully arrived at quantitative descriptions of desired characteristics of tested models. However, due to the fact that few groups of training sets were used in the simulations, relatively wider confidence intervals were unavoidably produced.

In order to provide more reliable results with regard to the desired characteristics of models being tested, the minimum distance constraint (see section 2.3.1 in chapter 2) was employed. This constraint was not only the important seed that helped to derive very efficient procedure as described in chapter 2, but also the essential factor that enabled one to characterize memory performances in a real application aspect.

3. 2. Confidence Interval

To estimate the number of trials t_p (defined in section 2.3.3 in chapter 2), a number of preliminary tests were carried out before the actual investigation began. Results of these preliminary tests show that 400 training groups are sufficient for $\hat{P}_r$ to achieve approximately 95% confidence within the interval $[\hat{P}_r-0.05, \hat{P}_r+0.05]$. Here $\hat{P}_r$ denotes the estimated probability of the tested item calculated according to $\hat{P}_r$ = the total number of occurrences / total trials. One of these preliminary tests is described in the following example. In this example, the performance of accretive recall of the Hopfield network was tested. The network was configured as n=16, and the P_es, the probability of each bit being reversed in

input patterns, were set to 0 and 0.1. The number of test groups t_p was 400. Each test was repeated five times. Figure 3.1 shows that, for the testing procedure given in chapter 2,

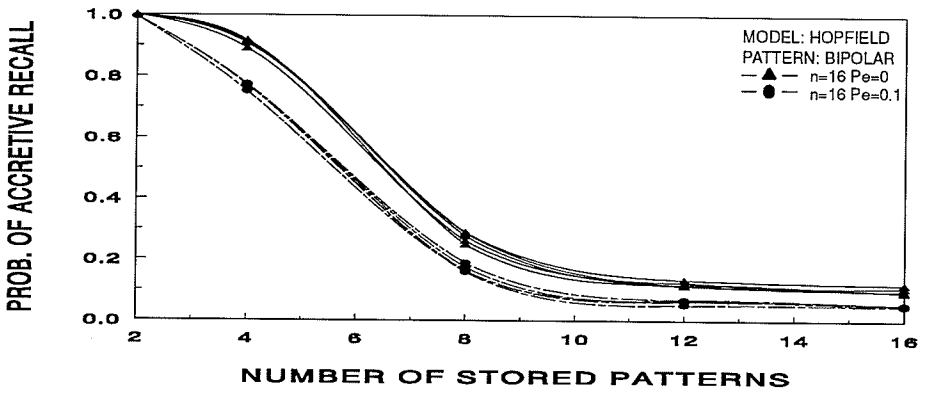

Fig. 3.1 Results of accretive recall from testing the Hopfield network. The network is tested 5 times, and each time has 400 randomly generated training patterns (with minimum distance constraint).

with t_p set to 400, the maximum interval among all tested points ($p = 8$ and $p_e = 0$ in this case) with 95% confidence is approximately [$\overline{P}_r$–0.024. $\overline{P}_r$+0.024]. $\overline{P}_r$ which is equal to 0.268 is the probability of recalling stored patterns averaged over five independent tests. The confidence interval was calculated according to [Ross1987]

$$[\ \overline{P}_r - \xi,\ \overline{P}_r + \xi\ ] \qquad (3.1)$$

where

$$\xi = t_{0.05/2,z-1}\ \frac{s}{\sqrt{z}}$$

where $\overline{P}_r$ is sample mean of $\hat{P}_r$s, t denotes the t distribution with z–1 degrees of freedom [Ross1987], z is a sample size, and s refers to the sample standard deviation. For the purpose of estimating whether 400 groups of training sets were sufficient, additional test was

performed at the point which yielded maximum interval in the test described above, i.e., p=8 and P_e=0. This time, the number of groups of training patterns, t_p s, were from 25 to 400, with specific values being: 25 50, 100, 200, 300, and 400. For each t_p, the network was tested five times. The result of each performance was averaged over five tests and the upper and lower bound of confidence intervals were plotted in figure 3.2. It is shown that as t_p becomes larger than 200, the 95% confidence interval for this tested point has already narrowed to approximately [$\overline{P}_r$–0.032. $\overline{P}_r$+0.032]. In order to assure that all results fall within the desired confidence interval, the number of groups of training sets, t_p , was set to 400 throughout the following experiments. It is necessary to point out that Eqn. (3.1) is derived under the

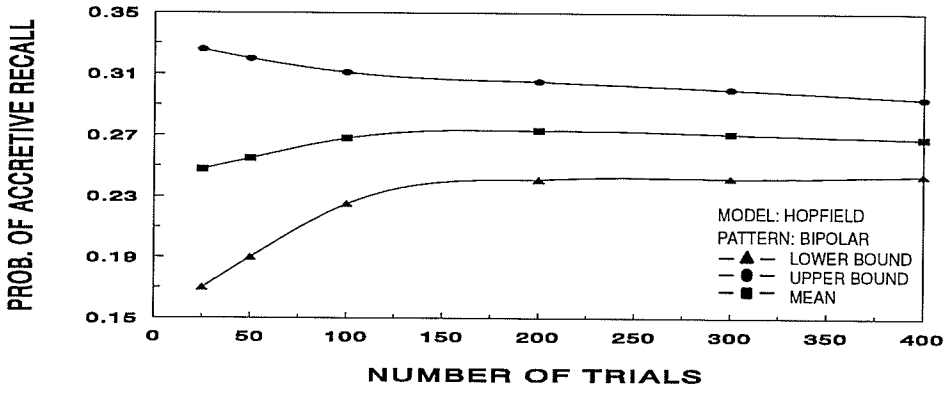

Fig. 3.2 Confidence interval of accretive recall. Tested model: Hopfield network. Network configuration: n=16. The number of stored patterns: p=8. Input noise: the probability of each bit being reversed in input pattern, P_e, is set to 0.

assumption that z is sufficiently large and the $\hat{P}_r$s should be normal random variables. In practice, however, it is very difficult to produce a large z as well as to verify whether or not $\overline{P}_r$ is normal. For this reason, the confidence interval, [$\hat{P}_r$–0.05, $\hat{P}_r$+0.05], is only

approximate in terms of coverage.

3. 3. Hopfield Network – an Autoassociative Memory

In 1982, Hopfield introduced a vigorous kind of associative memory based on his studies of collective computation. The network is composed of a single layer of neurons which are fully interconnected with each other. The strength of connections set during the encoding process provides the global communication of information. The strong nonlinearity of logical function implanted in each neuron enables the network to accomplish many sophisticated tasks such as making choices, producing categories, regenerating information, and performing nearest neighbor searches. Thus, in spite of the simplicity of the highly formalized neural structure, considerable network computation capability is intrinsic in the system [MPRV1987].

3. 3. 1 Network structure

A fundamental Hopfield network is a set of simple bistable elements, each of which is capable of assuring two values: +1 (firing) and –1 (nonfiring). The state of each neuron, +1 or –1, then represents a bit of information, and states of network delineated by n–tuple bipolar patterns (provided there are n neurons in the network) represent the entire information stored in the memory. It is necessary to note that the network is also able to handle a unipolar mode, {0, 1}, but a bipolar mode, {–1, 1}, must be used to encode information. Furthermore, the nonlinear transform function must be replaced by the unit step function (see Fig. 1.4 in chapter 1) if unipolar patterns are used to recall stored information. Because the unipolar

mode only participates in the recall phase, only the bipolar mode is discussed in this section. Further, the network addressed here is assumed as fully interconnected through linear

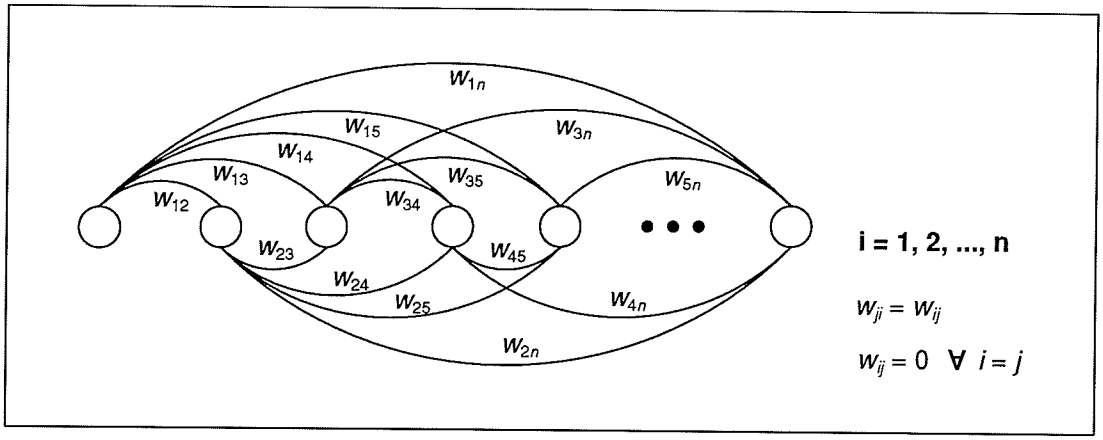

Fig. 3.3 Structure of Hopfield network.

synaptic connections w_{ji} transmitting a bit of information from neuron i to another neuron j. The weight matrix is symmetric, i.e., $w_{ji} = w_{ij}$, and with zero in its diagonal. The structure of Hopfield network is depicted in Fig. 3.3.

3. 3. 2 Encoding algorithm

There are two questions which may be raised naturally: First, how can the network store information so densely in weights? Second, how can the network recall the most similar pattern thereafter? A brief answer to these questions can be only one word: "learning". One simple but quite efficient learning algorithm is the Hebbian learning rule [Hebb1960]. This idea first came out of the similarity between, and the hypothesis about, the way that synaptic strength in the brain changes in response to experience [HeKP1991]. It says that the information in the human brain is stored by changing the strength of synapses and such a

change must be proportional to the connection between firing of the pre– and post–synaptic neurons [Hebb1960]. This hypothesis can be formulated as

$$\Delta w_{ji} = \eta\, x_j^{(s)}\, x_i^{(s)} \tag{3.2}$$

which means that synaptic weights are updated based on the multiplication of information taking on neuron i and j. In the learning process, Eqn. (3.2) actually identifies how much attention needs to be paid to particular neurons i and j. If neuron i and neuron j are both firing (or nonfiring) the connection strength between them should be increased, otherwise it ought to be decreased. An alternative way to express equation (3.2) is using an adaptive form which seems to be more applicable for a software implementation:

$$w_{ji(t)} = w_{ji(t-1)} + \eta\, x_j^{(s)}\, x_i^{(s)} \tag{3.3}$$

here t is an iteration number. Further, if, a special case is considered where each pattern is presented only once, then

$$w_{ji} = \eta \sum_{s=1}^{p} x_j^{(s)}\, x_i^{(s)} \tag{3.4}$$

which is identical to the form of

$$w_{ji} = \sum_{s=1}^{p} x_j^{(s)}\, x_i^{(s)} \tag{3.5}$$

provided the learning rate η is set to 1.

Technically however, Eqn. (3.2) goes beyond Hebbian's original hypothesis because firstly, it changes the weight positively when neither of the neurons is firing. This is probably

not physiologically reasonable. Secondly, Eqn. (3.2) can even cause a particular weight to change from excitatory to inhibitory or vice versa as more patterns are added. This phenomenon is hard to believe to occur at real synapses [HeKP1991].

3. 3. 3 Associative recall

For a neuron j in the system, the activation x_j at one particular moment is defined as

$$x_j = sgn\left(\sum_{i=1}^{n} w_{ji}\, x_i - \gamma_j\right) . \tag{3.6}$$

Eqn. (3.6) describes the evolution strategy of the network dynamics. The threshold decision rule (the sign function or hardlimiter) which is illustrated in Fig. (1.4) in chapter 1 furnishes two operation values, +1 and –1. The computation is quite simple: first of all, a neuron evaluates the weighted sum of the bipolar states of all other neurons asynchronously (or synchronously) in the network. The new state of the neuron is –1 if the sum is less than the threshold, and +1 if the sum exceeds the threshold. The neuron keeps its old value if the weighted sum is equal to the threshold. Such a nonlinear decision rule is expressed as

$$x_j = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} w_{ji}\, x_i > \gamma_j \\ unchange & otherwise \\ -1 & \text{if } \sum_{i=1}^{n} w_{ji}\, x_i < \gamma_j \ . \end{cases} \tag{3.7}$$

The Hopfield network is also an example of a feedback system. If input in the initial state is, for instance, an incomplete or noise corrupted picture, the network will search for a stable

state through several feedback cycles. A final state is the item which best matches the pattern according to what has been stored (though there is no guarantee in all cases for a system to converge to a correct state especially when memory is overloaded). The question remaining is what force enables the initial state in the memory to move to the desired point in the system configuration space and then stay there firmly? The energy function introduced by Hopfield in 1982 [Hopf1982] makes this question extremely clear.

For the network described above, the energy function, E, has the following form:

$$E = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ji}\, x_i\, x_j + \sum_{i=1}^{n} \gamma_i\, x_i \tag{3.8}$$

where w_{ji} is the strength of connection between neuron i and j (w_{ji}=w_{ij}), x_i stands for a bit of information in neuron i, and γ_i refers to a threshold in the transfer function. Note that nothing is contributed to energy E when $i = j$, because $w_{ji} = 0$ for all $i = j$. Eqn. (3.8) shows that the energy of a given state is a function of the weights. This helps to explain why information can be stored in the Hopfield network through proper selection of synaptic weights. On the other hand, Eqn. (3.8) suggests that the energy function also depends on what kind of information is represented. Therefore, one can imagine the surface of Eqn. (3.8) as an energy landscape in the corners of a hypercube. Fig. (3.3) is an illustration of an energy landscape in a three dimensional perspective.

In Fig. (3.3), the x–y plane represents the 2^n corners of the hypercube and the z axis measures the energy potential. It can be seen that the landscape contains rich hills and valleys. If memorized patterns are assumed to be in the location of energy wells which have

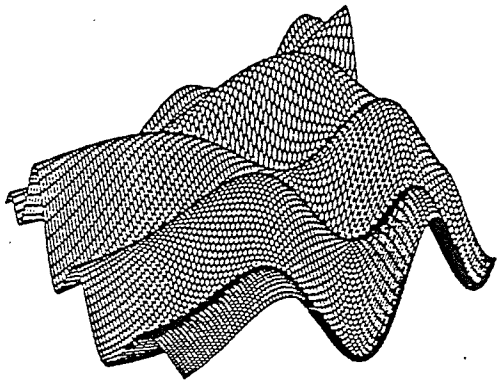

Fig. 3.4. Representation of the energy function in a three dimensional perspective. The lower Z value in the figure denotes the energy well where stable states are located. Source: Hertz, et al., Copyright by Addison–Wesley, Redwood City, CA, 1991.

relatively lower gravity potential, and the evolution strategy of the network dynamics can always ensure all kinds of initial patterns moving down the hill in the direction that decreases their potential, then the system is guaranteed to reach local energy minimum where full or complete information is located.

The central property of the energy function is that the energy potential either decreases or remains constant as the system evolves according to its dynamic adaptation rule given in Eqn. (3.6). To understand how the Hopfield network accomplishes associative recall, one can consider the case that the activation of neuron x_k, $1 \leq k \leq n$, has just been adapted from $x_k^{(old)}$ to $x_k^{(new)}$. Then, substituting into the energy function given in Eqn. (3.8) yields

$$E\left(x_k^{(new)}\right) = -x_k^{(new)} \sum_{i=1}^{n} w_{ki}\, x_i - \frac{1}{2} \sum_{\substack{j=1 \\ j \neq k}}^{n} \sum_{\substack{i=1 \\ i \neq k}}^{n} w_{ji}\, x_j\, x_i + \gamma_k\, x_k^{(new)} + \sum_{\substack{j=1 \\ j \neq k}}^{n} \gamma_j\, x_j \tag{3.9}$$

$$E\left(x_k^{(old)}\right) = -x_k^{(old)} \sum_{i=1}^{n} w_{ki}\, x_i - \frac{1}{2} \sum_{\substack{j=1 \\ j \neq k}}^{n} \sum_{\substack{i=1 \\ i \neq k}}^{n} w_{ji}\, x_j\, x_i + \gamma_k\, x_k^{(old)} + \sum_{\substack{j=1 \\ j \neq k}}^{n} \gamma_j\, x_j \tag{3.10}$$

The changed energy $\Delta E(x_k)$

$$\Delta E(x_k) = E\left(x_k^{(new)}\right) - E\left(x_k^{(old)}\right) \tag{3.11}$$

$$= -(x_k^{(new)} - x_k^{(old)}) \sum_{i=1}^{n} w_{ki}\, x_i + (x_k^{(new)} - x_k^{(old)})\, \gamma_k \tag{3.12}$$

$$= -\Delta x_k \sum_{i=1}^{n} w_{ki}\, x_i + \Delta x_k\, \gamma_k \tag{3.13}$$

$$= -\Delta x_k \left\{ \sum_{i=1}^{n} w_{ki}\, x_i - \gamma_k \right\} . \tag{3.14}$$

Because

$$x_k^{(new)} = sgn \left\{ \sum_{i=1}^{n} w_{ki}\, x_i - \gamma_k \right\} , \tag{3.15}$$

hence

$$\Delta E(x_k) = -\Delta x_k\, x_k^{(new)}\, C_k \tag{3.16}$$

where C_k represents the absolute value of the weighted sum of neuron k. Evidently, Δx_k must be either –2 or +2 because x_k is bipolar. There are three cases that need to be discussed:

case 1: $\Delta x_k = 0$. The energy is obviously unchanged.

case 2: $\Delta x_k = -2$. By Eqn. (3.15) $x_k^{(new)} = -1$, thus $\Delta E(x_k) < 0$.

case 3: $\Delta x_k = +2$. This is equivalent to $x_k^{(new)} = 1$. hence, $\Delta E(x_k) < 0$.

It is easy to show that energy is lower bounded at a specific value. This is because each component in the network has only two possible states +1 and –1 and the threshold, γ_i , must be within the range $-|\sum_{i=1}^{n} w_{ji}x_i| < \gamma_j < |\sum_{i=1}^{n} w_{ji}x_i|$. Therefore, the lower boundary for Eqn. (3.8) equals

$$-\frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{n}|w_{j,i}| - \sum_{j=1}^{n}\sum_{i=1}^{n}|w_{ji}| = -\frac{3}{2}\sum_{j=1}^{n}\sum_{i=1}^{n}|w_{ji}| . \tag{3.17}$$

Thus the energy decreases every time or remains constant as x_i, $i \in \{ 1, 2, ..., n \}$, changes, as claimed.

3. 3. 4 Experimental results

The performance of the Hopfield network was tested using the procedure described in chapter 2. Tested models contained 16 neurons. Each neuron was connected to other neurons through weights w_{ji}, where $i, j = 1, 2, ..., 16$. The neurons were not connected to themselves, i.e., $w_{ji} = 0$ for all $i = j$. In order to test the information capacity, the number of training patterns, p, stored in memories followed the sequence, 2, 4, 8, 12, 16, ..., until the performance, characterized by the probability of associative recall, drops down to the specified criterion, $P_{criterion}$=0.01. For each fixed p, there were 400 groups of training sets. Patterns in the training sets were randomly generated under the minimum distance constraint (see section 2.3.1 in chapter 2). The value of minimum distance, MD', is a function of p, the number of patterns to be stored in the memories. MD' was calculated according to

$$MD' = \frac{1}{Hd}\sum_{i=1}^{Hd} i \tag{3.18}$$

where Hd denotes the maximum mutual Hamming distance in the patterns subject to that the specified number of patterns, p', that can be generated. The relationship between Hd and the number of patterns, p', that can be generated is shown in Table 3.1 and Table 3.3. Two aspects must be clarified in order to calculate MD: (i) Because of the discontinuous nature in the relationship between Hd and p', a further decision for the choice of MD must be made. For instance, there are two groups of p'=32 patterns having mutual Hamming distance, Hd=7 and

Table 3.1 The Relationship Between Hd and p_{max}

Hd	The number of patterns *p'*
7	32
8	32
9	4
10	4
11	2

Note: Hd is the maximum mutual Hamming distance in the patterns sujebct to that the number of patterns , p', can be generated. $n = 16$.

Table 3.2 The Values of Minimum Distance MD

p	*MD'*	*MD*
16	4	4
12	4.5	[4,5]
8	5	5
4	5.5	[5,6]
2	6	6

Note: MD' is calculated according to Eqn. (3.18). $n = 16$.

Table 3.3 The Relationship Between Hd and p_{max}

Hd	The number of patterns *p'*
3	16
4	16
5	4
6	4

Note: Hd is the maximum mutual Hamming distance in the patterns sujebct to that the number of patterns , p', can be generated. $n = 8$.

Table 3.4 The Values of Minimum Distance MD

p	*MD'*	*MD*
16	2	2
12	2.5	[2,3]
8	2.5	[2,3]
4	3	3
2	3.5	[3,4]

Note: MD' is calculated according to Eqn. (3.18). n=8.

Hd=8, respectively. Whenever this situation occurs, the lower value of Hd is selected for larger p in the calculation of MD'. (ii) Since the gap between two p's may be wide, it is sometimes very difficult to select MD if the number of training patterns is set between them.

In order to overcome this problem, the actual MD used in testing may be composed of two values. These two values determining the upper and lower bounds of Hamming distance are calculated according to truncating and rounding the MD', where MD' is calculated according to Eqn. (3.18)). Another reason for introducing this two–value MD is to partially avoid MD exactly equaling to $\frac{1}{2}n$, which results in all training patterns being mutually orthogonal (in bipolar case only). The minimum distance constraint can be thought of as a filter. Randomly generated vectors have to be bypassed through this interval (checking their mutual Hamming distance) before being selected as training patterns. Table 3.2 and Table 3.4 give the values of MDs used throughout the testing. One more case which has not been discussed in determining MD is that MD' is an integer and p' is exactly equal to the number of training patterns, p. In this case, whether to use single–value MD or to use two–value MD is determined by if MD' is exactly equal to $\frac{1}{2}n$. If $MD' = \frac{1}{2}n$, the two–value MD is deemed necessary and the lower boundary is given by $MD'-1$ provided it is permissible. It is worth noting that mentioning this special case is mainly for the completeness of discussing how to determine MD; however, it, has not occurred in this experiment.

The performance of the Hopfield network was tested with respect to the following aspects:

(1) *Information capacity*:

According to the definition given in chapter 2, the information capacity can be shown by plotting the probability of memory successfully retrieving stored patterns (either accretively or interpolatively) against the memory load p. The maximum p observed under the condition that the probability of associative recall reaches the specified criterion will be considered as the capacity of that memory model. In the analysis of the information capacity,

the performance criterion, the probability of memory successfully retrieving stored patterns, is set to 0.95 unless otherwise specified,. Fig. 3.5 illustrates the result of the Hopfield network in performing accretive recall.

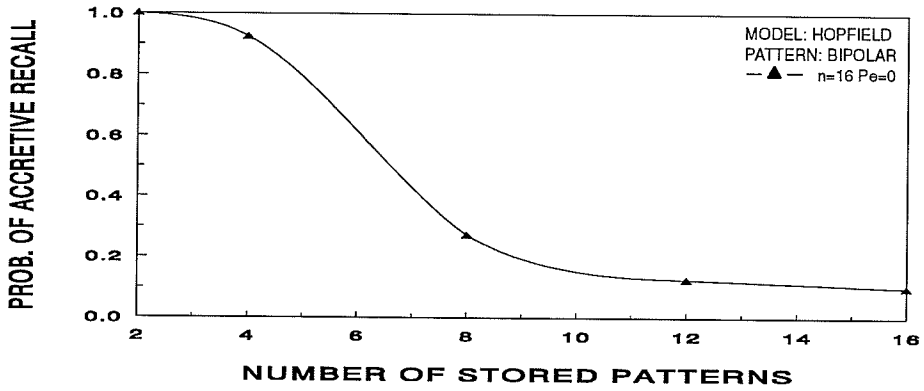

Fig. 3.5 Information capacity of the Hopfield network (accretive recall). The number of neurons: n=16. *Pe*, the probability of each bit being reversed in input patterns, equals 0.

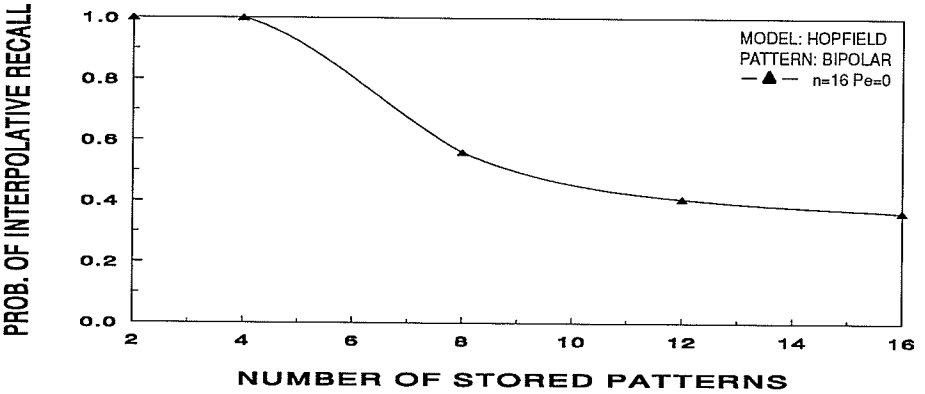

Fig. 3.6 Information capacity of the Hopfield network (interpolative recall). The number of neurons: n=16. *Pe*, the probability of each bit being reversed in input patterns, equals 0.

It is seen that the maximum pattern that can be stored in the tested Hopfield network is approximately 3. As to interpolative recall, the capacity remains at the same level as that of

accretive recall but the memory is guaranteed to recall all stored patterns with approximately 0.99 probability (Fig. 3.6). It is worth noting that the above experimental results do agree with the solution theoretically derived by McEliece et al. [MPRV1987]. According to the formula given by McEliece et al., an asymptotic capacity of the Hopfield network containing n neurons to get accretive recall (with absolutely no noise in input pattern) is less than

$$p < \frac{n}{2\log_2 n} . \tag{3.19}$$

Substituting 16 into Eqn. (3.19) yields

$$p < \frac{16}{2\log_2 16} \approx 3 . \tag{3.20}$$

Both (experimental and theoretical) results indicate that for the Hopfield network, the network configured as n=16 can store no more than four patterns.

(2) *Error correction capability*

Error correction capability was tested by varying noise level controlled by P_e but fixing stored pattern p. Because of the low capacity, the memory was unable to perform accretive recall perfectly if $p > 2$. The worst conditions under which the memory could still work well were p=2 and P_e=0.2. Under this condition, the memory was able to retrieve all stored patterns accretively with 0.95 probability. Thanks to the spurious states around the stored patterns (see Fig. 3.9 for details), the capability of correcting error was increased especially as many patterns are stored. The results of each kind of recall are shown in Figure 3.7 and in Fig 3.8.

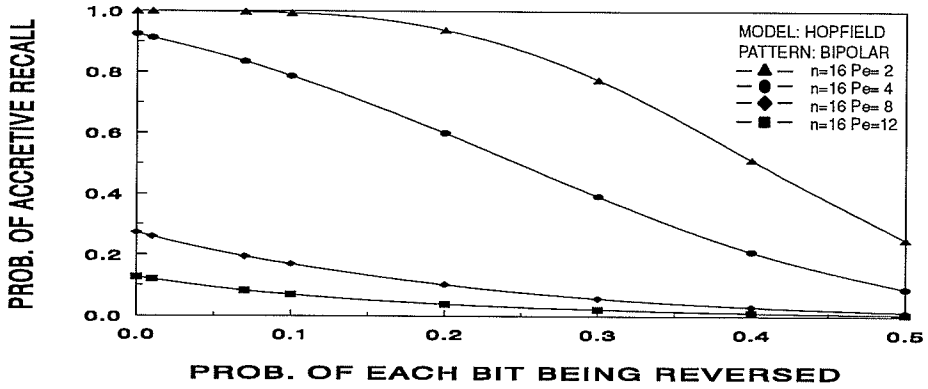

Fig. 3.7 Error correction capability of the Hopfield network (accretive recall). The number of neurons: n=16. Memory load: p = 2, 4, 8 and 12.

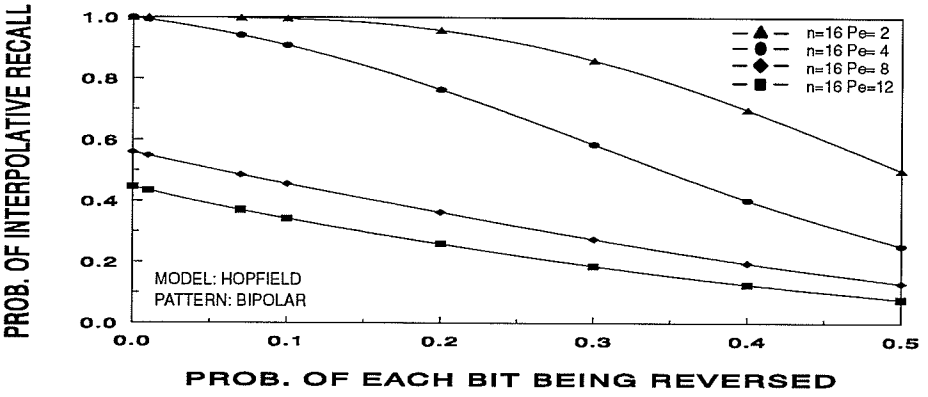

Fig. 3.8 Error correction capability of the Hopfield network (interpolative recall). The number of neurons: n=16. Memory load: p = 2, 4, 8, and 12.

(3) *Spurious and oscillatory states*

As stated in chapter 2, false states are composed of two disjointed sets: the spurious set and the oscillatory set. Spurious states were detected whenever memory performed neither accretive nor interpolative recall but converged to one of the stable states. Furthermore, these stable states should not be a member of the training sets. Test results show that the

probability for memories converging to spurious states (PCSS) is the function of the number of stored patterns. In general, the larger the number of patterns being stored, the higher the probability for a memory getting stuck at the spurious states. The results of PCSS are depicted in Fig. 3.9. It is necessary to point out that the value of PCSS varies if different P_es are chosen. For large P_e, the testing pattern may be far from the training pattern (in the terms of Hamming distance). As a result, the initial state has more chance being trapped into the spurious states. It has been shown that the spurious states in the Hopfield network are composed of the reversed version of training patterns and the Boolean combinations of training patterns [HaYo1989]. Fig. 3.9 also reveals that a large number of spurious states constitute false states. Few of them help the memory to increase the quality of interpolative recall.

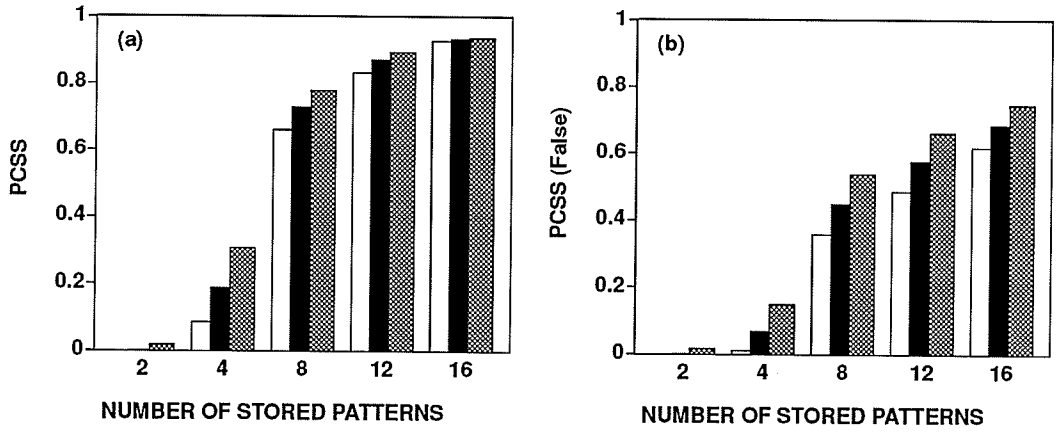

Fig. 3.9 (a): The probability of the Hopfield network converging to spurious states. (b): The probability of Hopfield network converge to false spurious states. Memory configuration: n=16. Input noise, the probability of each bit being reversed in input patterns, P_e is set to: □ P_e =0.01, ■ P_e =0.1, and ▩ P_e =0.2.

(4) *Oscillatory states*

Because an asynchronous recall strategy was employed throughout this experiment,

none of the oscillatory states was detected in the testing the Hopfield networks. The networks always converged to one of the stable states in the network configuration space.

3. 3. 5 Summary

Results presented here demonstrate that the Hopfield network can be used as autoassociative memory, but it only works well in storing a few patterns. The capability for a network to retrieve a stored pattern perfectly seems to be extremely difficult even when the network is evoked by noiseless inputs. Although the Hopfield network is inherent in some serious problems which seem to be very difficult to overcome, its simple structure and the suitability for VLSI implementation win it favor in many applications.

3. 4. Bidirectional Associative Memory (BAM)

Hopfield's seminal idea of simple neurons with symmetric connections behaving as if they minimize the energy in a physical system gives a firm analytical foundation for network computation [HiAn1989]. Since then, there has been much interest in developing associative memory by using neural network approaches. With the expansion of the single layer model, Kosko [Kosk1987], [Kosk1988] introduced a two–layer network called bidirectional associative memory (BAM) which was capable of performing both autoassociative and heteroassociative mapping tasks. Compared to the Hopfield network, BAM can achieve autoassociation with fewer weights. For instance, the Hopfield network needs $n \times n$ weights to store n–dimensional patterns while BAM only requires $n \times m (m < n)$ weights. On the other hand, BAM autoassociative mapping is accomplished by backward and forward mapping between two layers, thus the patterns chosen in the second layer are trivial. This allows for choosing the suitable training patterns in the second layer (i.e. as

sparse as possible, or to satisfy the continuous condition $1/n\ H(\mathbf{x}^{(s1)}, \mathbf{x}^{(s2)}) = 1/m\ H(\mathbf{y}^{(s1)}, \mathbf{y}^{(s2)})$ [Kosk1988]) in order to achieve higher performances.

3. 4. 1 BAM's evolution scheme

The primary goal of BAM is to retrieve a memory pair $(\mathbf{x}^{(s)}, \mathbf{y}^{(s)})$ given any one of initial input $\mathbf{x}'$ or $\mathbf{y}'$. Here $\mathbf{x}^{(s)}$ and $\mathbf{y}^{(s)}$ are in the vector form $\mathbf{x}^{(s)} = (x_1^{(s)}, x_2^{(s)}, \ldots, x_n^{(s)})^T$ and $\mathbf{y}^{(s)} = (y_1^{(s)}, y_2^{(s)}, \ldots, y_m^{(s)})^T$, s=1, 2, ..., p, respectively. The elements in $\mathbf{x}^{(s)}$ and $\mathbf{y}^{(s)}$ are assumed in a bipolar mode {–1, 1 }. Relatively higher performance of associative processing in BAM is credited to the dynamic bidirectional process approach. It has been mentioned that one of the most appealing features of associative memory is its ability to tolerate noise and/or partial input patterns, i.e., given an input pattern $\mathbf{x}'$ which is somewhat similar to the stored patterns $\mathbf{x}^{(s)}$, s=1, 2, ..., p, the memory will respond to its association $\mathbf{y}^{(s)}$ according to the transformation function

$$\mathbf{y}^{(s)} = \mathbf{T}_1(\mathbf{W}, \mathbf{x}^{(s)}), \qquad s=1, 2, \ldots, p. \tag{3.21}$$

In the feedforward network model, the transfer function $\mathbf{T}_1$ in Eqn. (3.21) takes on the whole responsibility to correct error. However, it is unusual for function $\mathbf{T}_1$ being able to fully accomplish this task, especially when the memory is loaded with a relatively larger number of associations or the input pattern is contaminated with serious noise. The evolution scheme utilized in BAM is analogous to that used in the Hopfield network except that neurons participating in the evolution are situated in different layers. In the Hopfield network, the next state of neuron x_i is determined by

$$x_i = sgn\left(\sum_{j=1}^{n} w_{ij}\, x_j - \gamma_j\right) \tag{3.22}$$

or simply expressed in a vector form

$$\mathbf{x}_{(t+1)}^{(s)} = \mathbf{T}\ (\mathbf{W}, \mathbf{x}_{(t)}^{(s)}) \tag{3.23}$$

where t is an iteration number. If $\mathbf{x}_{(t)}^{(s)}$ on the right hand of Eqn. (4.3) is replaced with $\mathbf{y}^{(s)}$ and suppress t+1 in $\mathbf{x}_{(t+1)}^{(s)}$ on the left hand term then

$$\mathbf{x}^{(s)} = \mathbf{T}\ (\mathbf{W}, \mathbf{y}^{(s)})\ . \tag{3.24}$$

Assigning number 2 to transform function **T** and **W** in Eqn. (3.24) respectively and combining Eqn. (3.24) with Eqn. (3.21) gives

$$\mathbf{y}_{(t+1)}^{(S)} = \mathbf{T}_1\ (\mathbf{W}_1, \mathbf{T}_2\ (\mathbf{W}_2, \mathbf{y}_{(t)}^{(s)})) \tag{3.25}$$

or

$$\mathbf{x}_{(t+1)}^{(S)} = \mathbf{T}_2\ (\mathbf{W}_2, \mathbf{T}_1\ (\mathbf{W}_1, \mathbf{x}_{(t)}^{(s)})) \tag{3.26}$$

Eqn. (3.25) and (3.26) exhibit the evolution strategy of BAM.

3. 4. 2 The nature of bidirectional search process

Topologically, BAMs are composed of two unidirectional heteroassociative memories. These two memories are connected in a closed loop so that short term memory can pass through one memory and then feedback through the other memory. Unlike unidirectional associative memories the concept of input and output neuron is blurred. Neurons which are

responsible for sending and receiving information are all treated as visible neurons though they work in two different vector spaces. However, one may still refer to those neurons which are responsible for receiving initial pattern as input neurons and neurons in the other layer as output neurons. The basic structure of BAM is illustrated in Figure 3.10.

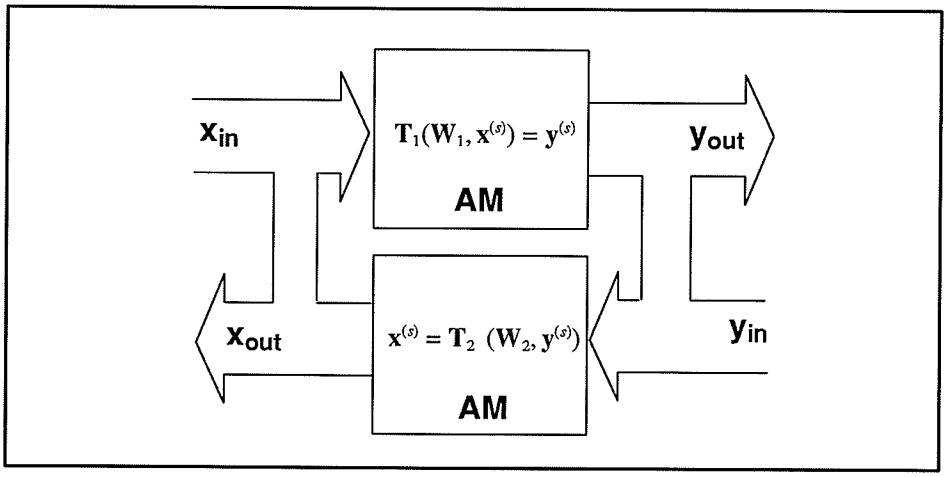

Fig. 3.10 Basic structure of BAM.

Figure 3.11 shows that BAM performs bidirectional mapping between two state spaces: the X space and Y space. The nature of a bidirectional operation can be captured by placing oneself between two state spaces and observing state transition simultaneously. To see how the dynamic bidirectional memory works, consider the case that BAM is loaded with p associations $\{\mathbf{x}^{(s)}, \mathbf{y}^{(s)}\}$, s=1, 2, ..., p. A dynamic bidirectional process begins as soon as the BAM is activated by a noise input pattern in X domain. First of all, initial state $\mathbf{x}$ evokes $\mathbf{y}'$ according to Eqn. (3.21). Though transform function, $\mathbf{T}_1$, does show some aptitude for correcting noise, it is unable to suppress the noise completely. Therefore, $\mathbf{y}'$ is unstable

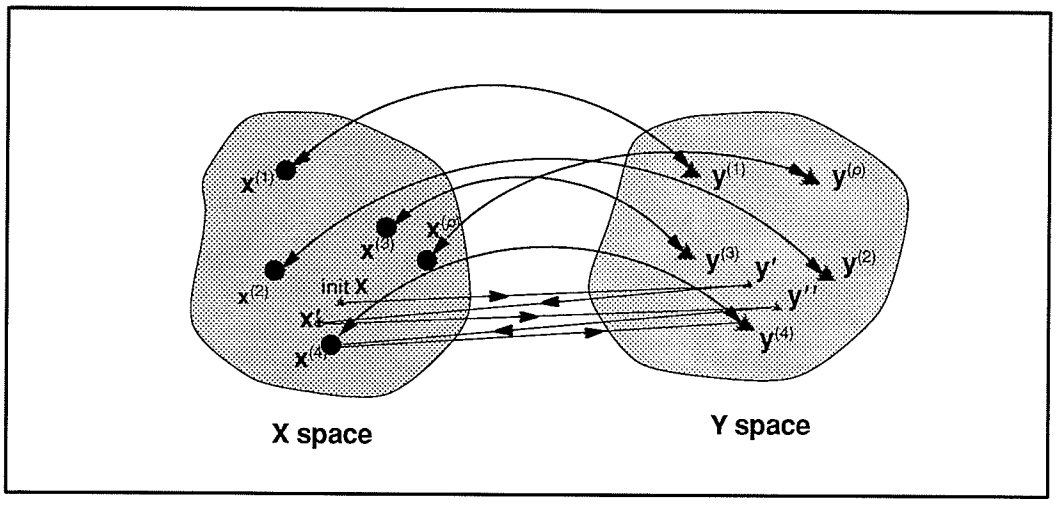

Fig. 3.11 Illustration of bidirectional recall in BAM.

and is forced to feedback to the X space through the other transform function, $\mathbf{T}_2$, generating $\mathbf{x}'$. Such back and forth transition continues until, ideally, BAM reaches $\{\mathbf{x}^{(4)}, \mathbf{y}^{(4)}\}$, one of the closest training pairs in $\{\mathbf{x}^{(s)}, \mathbf{y}^{(s)}\}$, s=1, 2, ..., p, from its starting state. In view of this example, the nature of dynamic bidirectional recall is merely a nonlinear feedback searching process. The recall always ends with memory converging to a stable state. The bidirectional process is readily formulated by

$$y_j = sgn\left(\sum_{i=1}^{n} w_{ji}\, x_i - \theta_j\right) \tag{3.27}$$

and

$$\mathbf{x}_i = sgn\left(\sum_{j=1}^{m} w_{ji}\, y_j - \gamma_i\right) \tag{3.28}$$

where sgn(.) is the hardlimiter activation function given in chapter 1. Combining Eqn. (3.27) and (3.28) yields

$$y_{j(t+1)} = sgn\left[\sum_{i=1}^{n} w_{ji} \;\bullet\bullet\bullet\; sgn\left(\sum_{j=1}^{m} w_{ji}\, y_{j(t)} - \gamma_i\right) - \theta_j\right]. \tag{3.29}$$

Such a chain of bidirectional search can be visually described as

$$\mathbf{x}' \xrightarrow{\mathbf{W}_1} \mathbf{y}' \xrightarrow{\mathbf{W}_2} \mathbf{x}'' \xrightarrow{\mathbf{W}_1} \mathbf{y}'' \bullet\bullet\bullet \longrightarrow \mathbf{x}^{(k)} \xrightarrow{\mathbf{W}_1} \mathbf{y}^{(k)} \xrightarrow{\mathbf{W}_2} \mathbf{x}^{(k)}$$

Unfortunately, there is no guarantee that BAM can always arrive at the desired point, the nearest stored association $\mathbf{x}^{(k)}$ and $\mathbf{y}^{(k)}$, from its starting point in the state space. The BAM is more likely to get stuck in spurious states.

3. 4. 3 Network stability

In order to get a better understanding of how the bidirectional associative memory works, it is necessary to review the property of convergence of BAM. This property was proposed and proved by Kosko [Kosk1988]. BAM's energy for a particular state, $(\mathbf{x}, \mathbf{y})$ is defined as

$$E(\mathbf{x},\mathbf{y}) = -\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ji}\, x_i\, y_j + \sum_{j=1}^{m} \theta_j\, y_j + \sum_{i=1}^{n} \gamma_i\, x_i\,. \tag{3.30}$$

Partitioning x_k, one of the activation in neuron $k, k \in \{1, 2, \ldots, n\}$, from terms on the right hand of Eqn. (3.30) will give

$$E(\mathbf{x},\mathbf{y}) = -\left[\sum_{\substack{i=1\\ i\neq k}}^{n}\sum_{j=1}^{m} w_{ji}\, x_i\, y_j + x_k \sum_{j=1}^{m} w_{ji}\, y_j\right] + \sum_{\substack{i=1\\ i\neq k}}^{n} \gamma_i\, x_i + \gamma_k\, x_k + \sum_{j=1}^{m} \theta_j\, y_j\,. \tag{3.31}$$

If a similar method is applied to y_l, $l \in \{1,2,\ldots,m\}$, then

$$E(\mathbf{x},\mathbf{y}) = -\left[\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq l}}^{m} w_{ji}\, x_i\, y_j + y_l \sum_{i=1}^{n} w_{ji}\, x_i\right] + \sum_{i=1}^{n} \gamma_i\, x_i + \theta_l\, y_l + \sum_{\substack{j=1\\ j\neq l}}^{m} \theta_j\, y_j\,. \tag{3.32}$$

According to Eqn. (3.32) the changed energy, $\Delta E = E^{(new)}_{(\mathbf{x},\mathbf{y})} - E^{(old)}_{(\mathbf{x},\mathbf{y})}$, caused by the state change in y_l, $\Delta y_l = y_l^{(new)} - y_l^{(old)}$, is

$$\Delta E = E^{(new)}_{(\mathbf{x},\mathbf{y})} - E^{(old)}_{(\mathbf{x},\mathbf{y})} = -(y_l^{(new)} - y_l^{(old)}) \left(\sum_{i=1}^{n} w_{ji}\, x_i - \theta_l\right). \tag{3.33}$$

Because y_l is in bipolar mode $\{-1,+1\}$, if $\Delta y_l > 0$ then $y_l^{(new)}$ must be equal to 1. In this case, the inequality

$$\sum_{i=1}^{n} w_{ji}\, x_i - \theta_l > 0 \tag{3.34}$$

must hold since according to equation (3.27), the left hand of Eqn. (3.34) is the netinput of $y_l^{(new)}$. If $\Delta y_l < 0$ then $y_l^{(new)}$ can only be –1 and the inequality

$$\sum_{i=1}^{n} w_{ji}\, x_i - \theta_l < 0 \tag{3.35}$$

holds. The above two cases result in $\Delta E < 0$. From equation (3.33) it is seen that $\Delta E = 0$ if and only if $\Delta y_l = 0$. This must be the final situation that BAM encounters. A similar situation would happen in the eqn. (3.31) if x_k changes state. Note that the energy value of BAM is bounded at

$$Min\ (E(\mathbf{x}, \mathbf{y})) = -3 \sum_{i=1}^{n} \sum_{j=1}^{m} |w_{ji}| . \tag{3.36}$$

This is because $-|\sum_{i=1}^{n} w_{ji}| < \theta_j < |\sum_{i=1}^{n} w_{ji}|$ and $-|\sum_{j=1}^{m} w_{ji}| < \gamma_i < |\sum_{j=1}^{m} w_{ji}|$. From the above discussion, it can be deduced that starting from any initial condition, BAM always converges to a local minimum. It should be noted that the necessary and sufficient condition for BAM to converge to stable states is that the weight matrix is symmetric, i,e., $w_{ij} = w_{ji}$, i=1, 2, ..., n, and j=1, 2, ..., m.

3. 4. 4 BAM's encoding algorithm

A fundamental BAM encoding is based on correlation matrix summation. Suppose there are p pairs of associations $\{\mathbf{x}^{(s)}, \mathbf{y}^{(s)}\}$, s=1,2,...,p, where $\mathbf{x}^{(s)}$ and $\mathbf{y}^{(s)}$ are column vectors respectively, then the weight matrix $\mathbf{w}_1$ is formed according to

$$\mathbf{W}_1 = (\mathbf{x}^{(1)})^T\mathbf{y}^{(1)} + (\mathbf{x}^{(2)})^T\mathbf{y}^{(2)} + \ldots + (\mathbf{x}^{(p)})^T\mathbf{y}^{(p)} = \sum_{s=1}^{p} (\mathbf{x}^{(s)})^T\mathbf{y}^{(s)} . \tag{3.37}$$

It can be seen that the encoding algorithm is still based on the hypothesis proposed by D. O. Hebb [Hebb1960] because the form of Eqn. (3.37) can readily be changed to

$$w_{ji} = \sum_{s=1}^{p} x_i^{(s)} y_j^{(s)} \tag{3.38}$$

where w_{ji} is the weight between neuron j and i, and it is the element in i_{th} row and j_{th} column in the weight matrix, $\mathbf{w}_1$, given in equation (3.37). Eqn (3.37) or (3.38) provides only one of the algorithms to synthesize weights and $\mathbf{W}_1$ performs transformation from the state in X space to the state in Y space only. For BAM, it needs another group of weights to realize the backward transformation from Y space to X space. A similar Hebbian encoding scheme is applied to the formation of $\mathbf{W}_2$.

$$\mathbf{W}_2 = (\mathbf{y}^{(1)})^T\mathbf{x}^{(1)} + (\mathbf{y}^{(2)})^T\mathbf{x}^{(2)} + \ldots + (\mathbf{y}^{(p)})^T\mathbf{x}^{(p)} = \sum_{s=1}^{p} (\mathbf{y}^{(s)})^T\mathbf{x}^{(s)} . \tag{3.39}$$

Because

$$\mathbf{W}_2 = \sum_{s=1}^{p} (\mathbf{y}^{(s)})^T\mathbf{x}^{(s)} = \left(\sum_{s=1}^{p} (\mathbf{x}^{(s)})^T\mathbf{y}^{(s)} \right)^T = \mathbf{W}_1^T , \tag{3.40}$$

the connection matrix $\mathbf{W}_2$ can easily be derived by simply transposing $\mathbf{W}_1$.

3. 4. 5 Experimental results

The preceding subsection shows how Kosko's BAM model performs associative recall by using bidirectional search. The analysis of the encoding algorithm indicates that the encoding algorithm used in BAM still follows the Hebbian rule. Kosko estimates that BAM's capacity is $C_{ap} < \min(n, m)$ [Kosko1988], but K. R. Hasins, et al. indicate that the $n / (2log_2 n)$, the asymptotic capacity for Hopfield network, can also be applied to BAM

[HaYo1989]. This capacity is substantially lower then Kosko's original estimation especially if n and m is large. One of the objects of this subsection is to clarify these arguments.

Three different BAM models were involved in the testing. These systems were configured as *models I*: n=16, m=16, *model II:* n=16, n=8, and *model III*: n=8, m=16 in order to verify whether the capacity was bounded at input and output dimensions. The number of pairs of associations stored in BAM followed the sequence, 2, 4, 8, 12, 16, ..., until the memory performances, characterized by the probability of associative recall, met the terminating criterion, $P_{criterion}$=0.01. Each training pair was generated separately under the *MD* constraint (see section 2.3.1 in chapter 2). The noise patterns which were determined by P_es, the probability of each bit being reversed, were set to 0, 0.01, 0.07, 0.1, 0.2, 0.3, 0.4, and 0.5. In the testing, training patterns were firstly encoded by using Eqn. (3.38) generating $\mathbf{W}_1$, and then $\mathbf{W}_1$ was used to synthesize $\mathbf{W}_2$ by simply transposing $\mathbf{W}_1$. Each model was tested individually using the procedure given in chapter 2. The number of training groups was fixed at 400. The results are as follows:

(1) Information capacity:

The performance of information capacity was tested in both accretive recall and interpolative recall cases. Fig. 3.12 illustrates the results of accretive recall for all three tested models. The first two models (*models I* and *models II*) successfully achieved high probability of accretive recall for p=4. Compared to the Hopfield network (configured as n=16), the probability of accretive recall when a memory stored four associations was approximately identical. However, for BAM if $n < m$, (n=8 and m=16 in this case), the performance of

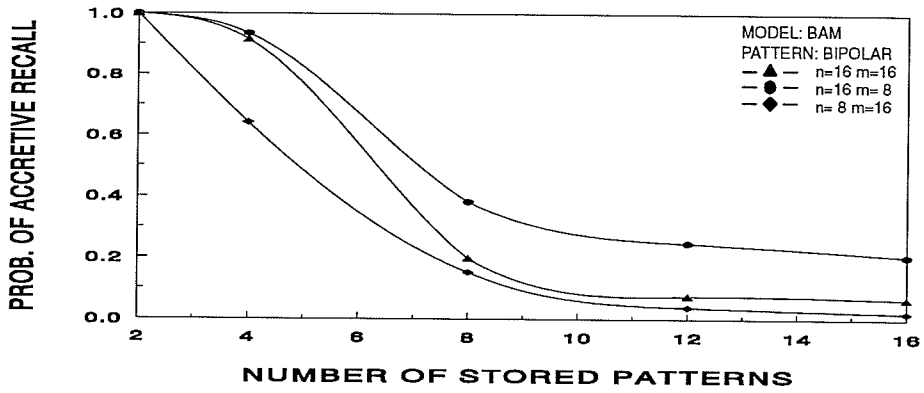

Fig. 3.12 Information capacity of BAM (accretive recall) for *Model I* (n=16, m=16.), *Model II* (n=16, m=8) and *Model III* (n=8, m=16). *Pe*, the probability of each bit being reversed in input patterns, equals 0.

accretive recall was degraded rapidly as the number of stored patterns increased. These results reveal that the information capacity of BAM is substantially lower than Kosko's estimation. The maximum pairs of associations that can be stored in this BAM is approximately 3. Results also show that the BAM constructed as n=16 and m=8 achieves relatively high performance, especially when the number of stored patterns exceeded the capacity limit.

The results of interpolative recall for the *model I*, n=16 and m=16, is depicted in Fig. 3.13. It is seen that the capacity is about 4. Unlike accretive recall, the performance of interpolative recall is less affected by input and output pattern dimensions. It almost remains at the same level even for $n < m$. For this reason their corresponding performances are omitted here.

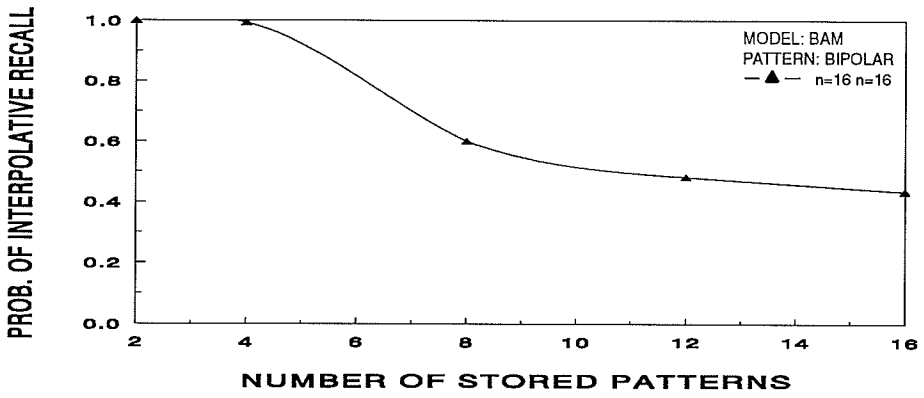

Fig. 3.13 Information capacity of BAM (interpolative recall) for *Model I*. Memory configuration: n=16, m=16. *Pe*, the probability of each bit being reversed in input patterns, equals 0.

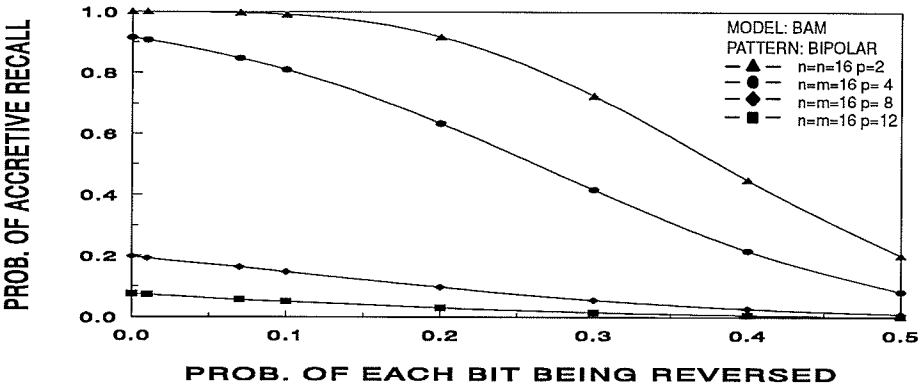

Fig. 3.14 The error correction capability of BAM (accretive recall). Memory configuration: n=16 and m=16. Memory load: p = 2, 4, 8, and 12.

(2) error correction capability:

The results of error correction capability of BAM is shown in Fig. 3.14 and Fig. 3.15. These results demonstrate that when storing two pairs of associations, the BAM is able to

get 0.92 probability of accretive recall even when P_e =0.2. BAM's error correction performance did not show much difference in both accretive and interpolative recall cases compared with the Hopfield network.

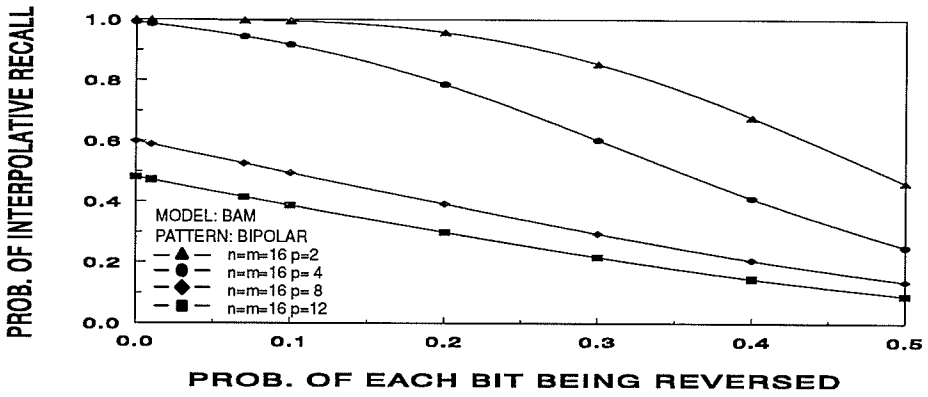

Fig. 3.15 The error correction capability of BAM (interpolative recall). Memory configuration: n=16 and m=16. Memory load: p = 2, 4, 8, and 12.

(3) *The effect of input and output pattern dimensions on accretive recall*

In order to test this property, one more BAM model was added. This model contained eight neurons in both input/output ports. Test results show that the input and output pattern dimensions only affects accretive recall. The effect of input and output pattern dimensions on other aspects is not obvious. Those results are omitted here. The effect on accretive recall was quantitatively calculated according to Eqn. (2.7) (refer to section 2.2.3 in chapter 2 for details) and the corresponding results are shown in Fig. 3.16.

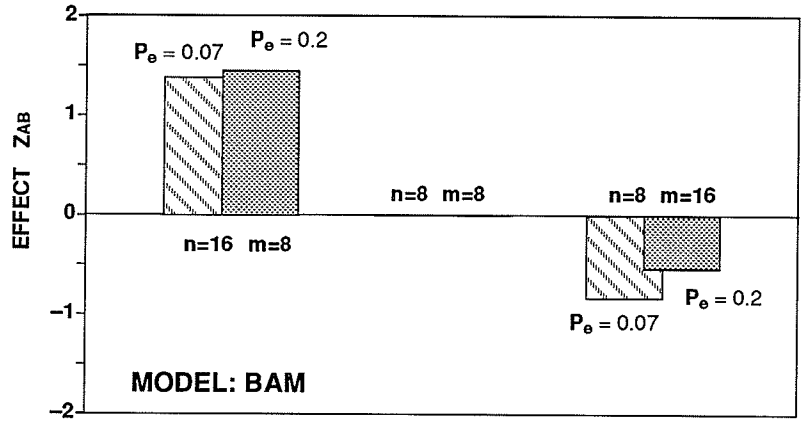

Fig. 3.16 The effect of input and output pattern dimensions on accretive recall. Positive value suggests the increase in the performance, and the negative value denotes the decrease in the performance. Here n and m are the dimensions of input and output patterns respectively. P_e stands for the probability of each bit being reversed in input patterns.

The test results indicate that if BAM is constructed as $n > m$, relatively higher performance can always be achieved in accretive recall.

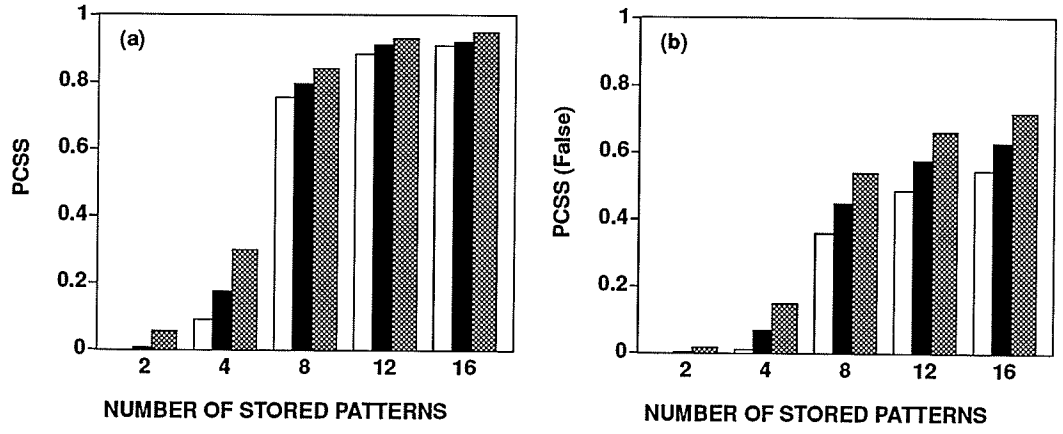

Fig. 3.17 (a): The probability of BAM converging to spurious states. (b): The probability of BAM converge to false spurious states. Memory configuration: n=16 and m=16. Input noise: the probability of each bit being reversed in input patterns, P_e is set to: □ P_e =0.01, ■ P_e =0.1, and ▩ P_e =0.2.

(4) *Spurious and oscillatory states*

No oscillatory case was detected in the testing of BAM. This result is consistent with the result of stability analysis proposed by Kosko [Kosk1988]. The performance of BAM converging to spurious states was also investigated. The probability for BAM being trapped into spurious states grows rapidly if the number of stored patterns exceeds the capacity limit, $n/(\ 2log_2 n\ )$. These results, as shown in Fig. 3.17, demonstrate that BAM, like the Hopfield network, suffers from large numbers of spurious states. Approximately 75% of spurious states constitute the false states.

3. 4. 6 Summary

This subsection presents quantitative descriptions about BAM's performance affected by different loads (the number of pairs of associations stored in the memory), the noise in input patterns, and network structures. In general, all performances remain at the same level as those of the Hopfield networks. Test results manifest that the information capacity for BAM is much lower than min(n, m). The upper bound for Hopfield network, $n/(\ 2log_2 n\ )$, can be directly applied to estimating BAM's capacity. Although BAM is bidirectional, and the neurons in either layer can be used as input or output, the performance is by no means identical. Test results show that relatively higher performance can be obtained if $n > m$ (here n is assumed as the same dimension as that of the input pattern). Another finding is that BAM's capacity is mainly determined by the dimension of input pattern. The dimension of output pattern is less important.

3. 5. Ho–Kashyap Associative Memory

In general, once the network architecture and training patten is given, overall memory performance will be determined by the encoding algorithm that is used [Hass1989]. In the previous chapter, two types of dynamic associative memories have been investigated. These memories simply use the Hebbian rule. Although this simple "one shot" encoding rule allows very large associative memory to be implemented in a simple chip and makes storage and removal of the contents of memory relatively easy, it suffers from low storage capacity and a large number of unwanted false states. In an attempt to avoid using a correlation type of the encoding scheme which had proved many inherent shortcomings, M. H. Hassoun proposed a dynamic bidirectional memory (HK model) [Hass1989]. Unlike BAM, the HK model utilizes nonsymmetric weights formalized individually during the encoding process. These weights are constructed by employing the Ho–Kashyap algorithm which optimally distributes the association process of each neural layer over groups of individual neurons and activation functions [Hass1989]. The high performance is contributed to the Ho–Kashyap learning algorithm being capable of making optimal use of a nonlinear activation function as a part of the recall process. However, in the Hopfield network and BAM, the similarity measure (the weighted sum of all other neurons' activations) is realized in the interconnection layers. The advantage of nonlinear thresholding, which is beyond in helping the stabilization of the memory, is not fully utilized. Thresholds used in these memories are obviously not the optimal choice. Consequently, they are not as essential as those in other memories such as encoding of information based on the Ho–Kashyap algorithm. Nevertheless, the superior performance of the HK model comes at the cost of both the increased temporal computation complexity and the doubled size of weights.

3. 5. 1 Mathematical background of Ho–Kashyap learning algorithm

For the purpose of capturing key points of the Ho–Kashyap associative memory encoding algorithm, it is necessary to review the fundamental Ho–Kashyap algorithm [HoKa1965], [SkWa1981]. Consider the problem of designing a classifier to generate a hyperplane that optimally separates two groups of linear separable feature vectors in the training set. To do this one may define a cost function that numerically summarizes the error of the performance:

$$E\ (\mathbf{w}, \mathbf{b}) = \sum_{s=0}^{p} (\ \mathbf{w}^T\ \mathbf{x}^{(s)} - b^{(s)}\ )^2 \tag{3.41}$$

where $\mathbf{x}^{(s)} = (x_1^{(s)}, x_2^{(s)}, \ldots, x_n^{(s)})^T$ stands for a feature vector, $\mathbf{w} = (w_1, w_2, \ldots, w_n)^T$ and scalar $b^{(s)}$ are controllable parameters that need to be adjusted. The object is to find a vector $\mathbf{w}$ as well as $b^{(s)}$ such that the sum–of–square–error $E(\mathbf{w}, \mathbf{b})$ is minimum. Equation (3.41) is readily changed in the matrix form. First of all, define $\mathbf{X}$ as a p by n matrix:

$$\mathbf{X} = \left[\ \mathbf{x}^{(1)},\ \mathbf{x}^{(2)},\ \ldots,\ \mathbf{x}^{(p)}\ \right]^T \tag{3.42}$$

and $\mathbf{b}$ as a p dimensional vector:

$$\mathbf{b} = \left[\ b^{(1)},\ b^{(2)},\ \ldots,\ b^{(p)}\ \right]^T \tag{3.43}$$

then the equation (3.41) can be modified as

$$\begin{aligned} E\ (\ \mathbf{w}, \mathbf{b}\ ) &= \|\ \mathbf{X}\ \mathbf{w} - \mathbf{b}\ \|^2 \\ &= (\mathbf{X}\ \mathbf{w} - \mathbf{b}\ )^T\ (\mathbf{X}\ \mathbf{w} - \mathbf{b}\ ) \end{aligned} \tag{3.44}$$

It is known that the Ho–Kashyap algorithm achieves the minimum sum–square–error solution of $E(\mathbf{w}, \mathbf{b})$ by iteratively adjusting both $\mathbf{w}$ and $\mathbf{b}$ according to gradient descent. Gradient descent means that each adaptation step taken on $\mathbf{w}$ and $\mathbf{b}$ should be in the direction against the gradient of $E(\mathbf{w}, \mathbf{b})$ (with respect to $\mathbf{w}$ and $\mathbf{b}$). Partially differentiating $E(\mathbf{w}, \mathbf{b})$ with respect to $\mathbf{w}$ and $\mathbf{b}$ yields:

$$\frac{\partial E(\mathbf{w}, \mathbf{b})}{\partial \mathbf{w}} = 2\, \mathbf{X}^T (\mathbf{X}\, \mathbf{w} - \mathbf{b}) \tag{3.45}$$

and

$$\frac{\partial E(\mathbf{w}, \mathbf{b})}{\partial \mathbf{b}} = -2\, (\mathbf{X}\, \mathbf{w} - \mathbf{b})\,. \tag{3.46}$$

The Ho–Kashyap algorithm is described in the following statements:

Ho–Kashyap algorithm [HoKa1965]

Step (1) At time t=0 set arbitrary vector $\mathbf{b}_{(0)} > 0$.

Step (2) For a fixed $\mathbf{w}$ allows $\mathbf{b}_{(t)}$ to change in the direction of steep descent subject to $\mathbf{b}_{(t)} > 0$, i.e.,, $\mathbf{b}_{(t+1)} = \mathbf{b}_{(t)} + \eta\, (\mathbf{e}_{(t)} - |\, \mathbf{e}_{(t)}\, |)$, where the error $\mathbf{e}_{(t)}$ is defined as $\mathbf{e}_{(t+1)} = \mathbf{X}\, \mathbf{w}_{(t)} - \mathbf{b}_{(t)}$.

Step (3) For a fixed $\mathbf{b}$ bring $\mathbf{w}$ to the least–square solution of for the matrix equation $\mathbf{X}\, \mathbf{w} = \mathbf{b} > 0$. This is achieved by computing the generalized inverse matrix $\mathbf{X}^+$ and calculating $\mathbf{w}_{(t)} = \mathbf{X}^+\, \mathbf{b}_{(t)}$.

Step (4) Go to step (2) until $\mathbf{b}_{(t+1)} = \mathbf{b}_{(t)}$, no further change takes place.

3. 5. 2 Complete Ho–Kashyap associative memory encoding algorithm

To apply Ho–Kashyap algorithm to encoding pairs of associations, several modifications need to be carried out. These modifications are included in the algorithm given below.

Special notations

W : A m by n weight matrix

$\mathbf{w}_j^*$: A j–th row vector in the weight matrix but augmented by the threshold, θ_j, in the output neuron j.

$$\mathbf{w}_j^* = \left[-T_j \; w_{j1} \; w_{j2} \; \ldots \; w_{jn} \right]^T = \left[w_{ji} \right]$$

$$i = 0, 1, 2, \ldots, n \qquad j = 0, 1, 2, \ldots, m \qquad w_{j0} = -\theta_j$$

X A p by n matrix $\mathbf{X} = \left[\mathbf{x}^{(1)}) \; \mathbf{x}^{(s)}) \; \ldots \; x^{(p)} \right]^T$ formed by n–dimensional input patterns $\mathbf{x}^{(s)}$, $\mathbf{x}^{(s)} = (x_1^{(s)}, x_2^{(s)}, \ldots, x_n^{(s)})^T$, s=1, 2, ... , p.

Y A p by m matrix $\mathbf{y} = \left[\mathbf{y}^{(1)}) \; \mathbf{y}^{(s)}) \; \ldots \; y^{(p)} \right]^T$ formed by m–dimensional output patterns $\mathbf{y}^{(s)}$, $\mathbf{y}^{(s)} = (y_1^{(s)}, y_2^{(s)}, \ldots, y_m^{(s)})^T$, s=1, 2, ... , p.

X′ A p by n+1 matrix whose s–th row $\mathbf{x}'^{(s)}$ given by the augmented input pattern, $\mathbf{x}^{(s)}$, according to the rule:

$$\mathbf{x}'^{(s)} = \begin{cases} [1 \;\; \mathbf{x}^{(s)^T}] & \text{if } y_j^{(s)} = 1, \\ [-1 \;\; \mathbf{x}^{(s)}] & \text{if } y_j^{(s)} = -1 \text{ or } 0 \end{cases}$$

To be more specific:

$$\mathbf{x}'^{(s)} = [\ 1\ x_1^{(s)}\ x_2^{(s)},\ \ldots,\ x_n^{(s)}] \quad \text{if } y_j^{(s)} = 1$$
$$= [a_{0s}\ a_{1s}\ a_{2s},\ \ldots,\ a_{ns}] \quad i = 0, 1, 2, \ldots, n$$
$$= [a_{is}]$$

$$\mathbf{x}'^{(s)} = [\ -1\ -x_1^{(s)}\ -x_2^{(s)},\ \ldots,\ -x_n^{(s)}] \quad \text{if } y_j^{(s)} = -1 \text{ or } 0$$
$$= [-a_{0s}\ -a_{1s}\ -a_{2s},\ \ldots,\ -a_{ns}]$$
$$= [-a_{is}] \quad i = 0, 1, 2, \ldots, n$$

Note: (i) $y_j^{(s)}$ is the j–th element in the output pattern $\mathbf{y}^{(s)}$, (ii) Ho–Kashyap encoding algorithm needs to compute m $\mathbf{X}'$s for different elements in the output pattern $\mathbf{y}^{(s)}$, $s = 1, 2, \ldots, p$.

$\mathbf{b}_j$ a p by 1 column vector : $\mathbf{b} = \left[\ b^{(1)}\ b^{(s)}\ \ldots\ b^{(p)}\ \right] = \left[\ b^{(s)}\ \right]$, s=1, 2, ..., p.

Ho–Kashyap encoding algorithm [Hass1989]

step (1) Initialize vector $\mathbf{b}_{(t)} > 0$.

step (2) Compute $\mathbf{X}'^{+}$, the generalized inverse of matrix $\mathbf{X}'$ (note that $\mathbf{X}'$ is p by n+1 matrix, thus $\mathbf{X}'^{+}$ is n+1 by p).

step (3) Compute j–th row in the weight matrix, $\mathbf{W}_1$, namely, $\mathbf{w}_{j(t)}^{*}$ at time t=0,

$$\mathbf{w}_{j(t)}^{*} = \mathbf{X}'^{+}\ \mathbf{b}_{(t)}$$

or
$$-\theta_j = \sum_{s=1}^{p} a_{0s}^{+}\ b_{(t)}^{(s)} \quad \text{and} \quad w_{ji} = \sum_{s=1}^{p} a_{is}^{+}\ b_{(t)}^{(s)}\ , \quad \forall\ i,\ \ i = 1, 2, \ldots, n.$$

It should be noted that (i) the weight matrix $\mathbf{W}$ is used to produce memory response, $\mathbf{y}_j = \left[\ y_j^{(1)}\ y_j^{(2)}\ \ldots\ y_j^{(p)}\ \right]$ when a memory is evoked by a particular pattern,

$\mathbf{x}^{(s)}$, during the associative recall, (ii) the augmented weight vector (the j–th row vector in weight matrix) is only used in an encoding process. The numerical value of an element, $y_j^{(s)}$, in the vector, $\mathbf{y}_j'$ ($\mathbf{y}'_j = \left[-\theta_j \; y_j^{(1)} \; y_j^{(2)} \; \ldots \; y_j^{(p)} \right]$ which is resulted from the multiplication of $\mathbf{y}_j' = \mathbf{X}' \; \mathbf{w}_j^*$) is a bit information generated by the output neuron j when a memory is evoked by the input pattern $\mathbf{x}^{(s)}$, except for the first element, $-\theta_j$, which is the threshold in the neuron j.

step (4) Calculate error: $\mathbf{e} = \mathbf{X}' \; \mathbf{w}^*_{j(t)} - \mathbf{b}_{(t)}$. if a symbol $[b_{(t)}^{(s)}]$ is used, then

$$e_{(t)}^{(s)} = \left(\sum_{i=0}^{n} a_{is} \; w^*_{ji(t)} \right) - b_{(t)}^{(s)} , \qquad s = 1, 2, ..., p.$$

step (5) Modify the vector $\mathbf{b}_{(t)}$ to minimize the error

$$\mathbf{b}_{(t+1)} = \mathbf{b}_{(t)} + \eta \; (\mathbf{e}_{(t)} - | \; \mathbf{e}_{(t)} \; |) \; .$$

step (7) Compute new weights again

$$\mathbf{w}^*_{j(t+1)} = \mathbf{X}'^{+} \; \mathbf{b}_{(t+1)} \; .$$

step (8) Compare

$$\mathbf{b}_{(t+1)} = \mathbf{b}_{(t)} \; .$$

If it is false go to step (4), otherwise, it means $\mathbf{w}_j^*$ is trained. In this case, select the next output neuron, i.e., j+1, followed by calculating new $\mathbf{X}'$ based on the information given by $\mathbf{y}_{j+1}^{(s)}$, s=1, 2, ... , p, and then, go to the very beginning, step (1), to train $\mathbf{w}_{j+1}^*$.

Continue to do this until all weight (row) vectors in matrix $\mathbf{W}_1$ are trained.

Note: The other weight matrix, $\mathbf{W}_2$, can be obtained by interchanging matrix $\mathbf{X}$ and $\mathbf{Y}$ in the above formulas.

The Ho–Kashyap encoding algorithm outlined above is based on the original idea proposed by Hassoun [Hass1989]. The initial vector $\mathbf{b}_{(0)}$ is based on the amount of knowledge one has about the asymptotic value of $\mathbf{b}_{(\infty)}$. To ensure the convergence, the choice of learning rate η must be within the interval (0, 1], [SkWa1981].

3. 5. 3 Experimental results

Similar testing procedure and the network structure used in investigating BAM were applied to HK models. However, because of the different encoding algorithms that were utilized, some changes in the experiment must be specified. First of all, unipolar patterns were used in both the encoding and recall processes. Thus, the transfer function in each neuron was replaced by the unit step function (see Fig. (1.4) in chapter 1). Secondly, since the HK model uses nonsymmetric weights, the weight size for the network is doubled compared with BAM. Finally, the HK encoding algorithm differs from the Hebbian rule in adaptively modifying the weight vector **w** and margin vector **b**, alternatively. To capture this property, the number of iterations was recorded and the average iterations for the HK model to converge to a global minimum is calculated. The performances of HK models were tested for their ability to store the maximum number of associations and to correct noise. The effect of input and output pattern dimensions on accretive recall was also tested. The memory being trapped into spurious and oscillatory states during the associative recall was also investigated.

(1) *Information capacity*

The maximum association p that can be stored in the HK model was individually tested on three different networks. These networks differed in their dimensions either in input or

in output. The first network was n=m=16, the second was n=16, m=8, and the third was n=8, m=16. The purpose of testing these different models was to see if the number of stored patterns in the HK model could exceed the number of neurons in either layer. The test environment was identical to that of testing BAM. The probability of HK models successfully recalling stored associations accretively against the number of stored patterns is plotted in Fig. 3.18. These results show that the upper bound of the information

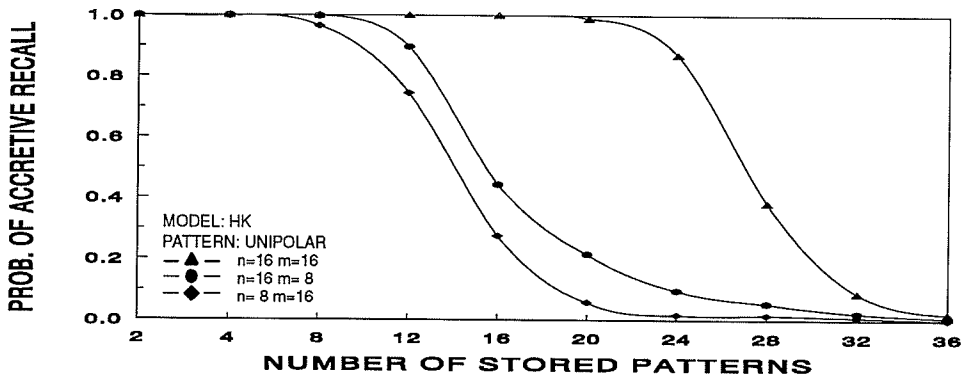

Fig. 3.18 Information capacity of the HK model (accretive recall) for *Model I* (n=16, m=16.), *Model II* (n=16, m=8) and *Model III* (n=8, m=16). *Pe*, the probability of each bit being reversed in input patterns, equals 0.

capacity for the HK model can be approximately estimated by calculating $\min(n, m)$ if $n \neq m$. It should be noted that this upper bound can only be reached if the memory is evoked by noiseless input. The result also demonstrates that if a memory is configured as $n = m$, the capacity is able to exceed its dimension. The capacity for the HK model configured as n=m=16 is approximately 23 which is approximately 1.4 times of the dimension of input or output pattern. This interesting phenomenon suggests that the dimension of input or output

pattern may not constitute the upper bound of the memory capacity. The actual limit of the information capacity largely depends on the encoding/learning algorithm used.

The performance of interpolative recall was also investigated. The results do not show much improvement in this respect. One conclusion made in accordance with this phenomenon is that few spurious states exist near the stored patterns; as a result, the memory either performs accretive recall or slips to the state far from the correct point.

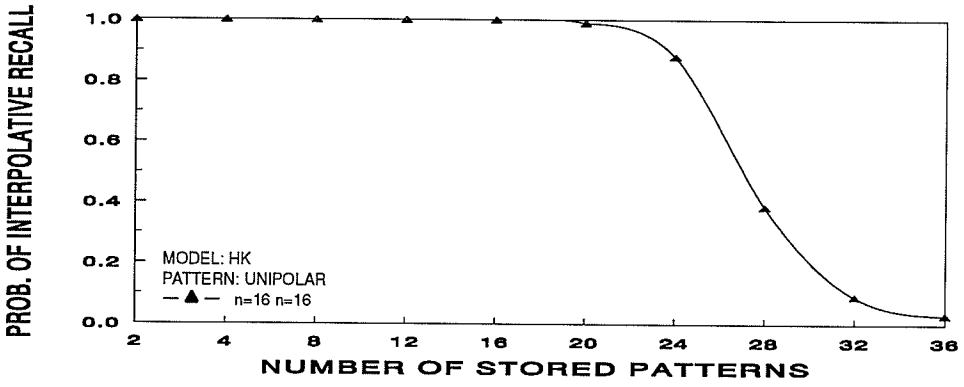

Fig. 3.19 Information capacity of the HK model (interpolative recall). Memory configuration: n=16, m=16. *Pe*, the probability of each bit being reversed in input patterns, equals 0.

(2) *error correction capability*

The ability of the HK model to recover distorted inputs was tested. Results for both accretive recall and interpolative recall are illustrated in Fig. 3.20 and 3.21. From these figures, one can see that the maximum noise level (determined by P_e) that the HK model (n=16, m=16) can tolerate for storing two associations is approximately 0.2. This result does not show any improvement compared to BAM. However, due to the increased information capacity, the HK model demonstrates relatively higher performance in storing four as well as eight associations.

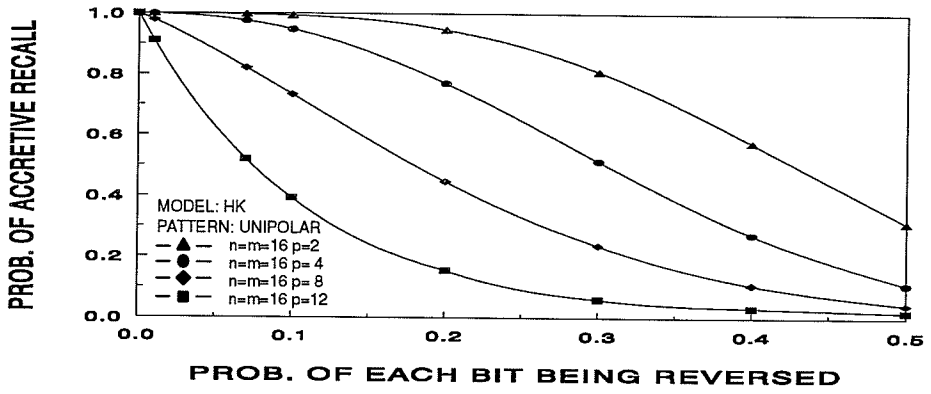

Fig. 3.20 The error correction capability of the HK model (accretive recall). Memory configuration: n=16 and m=16. Memory load: p = 2, 4, 8, and 12.

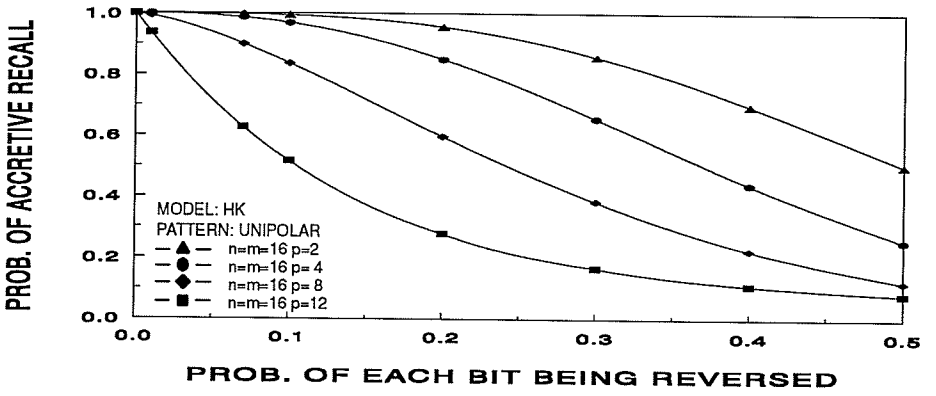

Fig. 3.21 The error correction capability of the HK model (interpolative recall). Memory configuration: n=16 and m=16. Memory load: p = 2, 4, 8, and 12.

(3) *The effect of input and output pattern dimensions on accretive recall*

The effect of input and output pattern dimensions on accretive recall was tested on three different HK models. Besides n=16, m=8 and n=8, m=16, the network configured as n=m=8 was also involved in the testing. This additional network was used as a reference model.

Results of two other models were compared to this reference model. This was done by calculating sum–square–root between the performance data measured under the following two different conditions: $n \neq m$ and $n = m$ (refer to section 2.2.3 in chapter 3 for details). The positive histogram shown in Fig. 3.22. means an increase in performance, whereas, the negative histogram stands for a decrease in performance. The degree of effect is represented by the height of the histogram. Although such an effect seemed to be not significant when

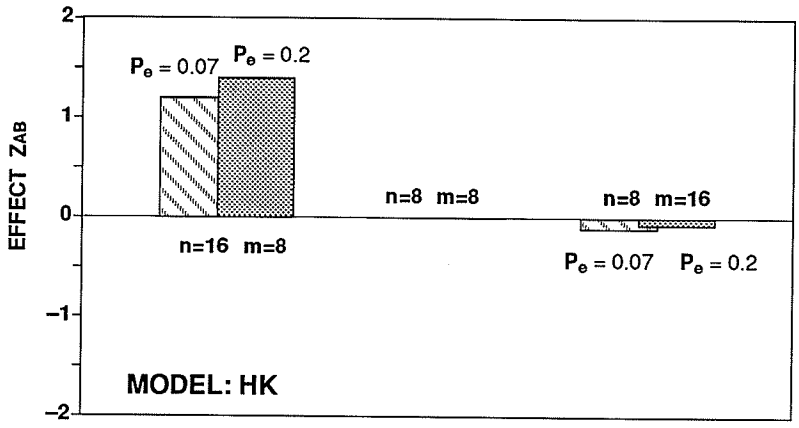

Fig. 3.22 (a): The effect of input and output pattern pattern dimensions on accretive recall. Positive value suggests the increase in the performance, and the negative value denotes the decrease in the performance. Here n and m are the dimensions of input and output pattens respectively. P_e stands for the probability of each bit being reversed in input patterns.

the number of associations stored in the memory is small, the effect became larger and larger as p increased. The overall effect (calculated according to Eqn. (2.7) in chapter 2) was less significant compared with BAM.

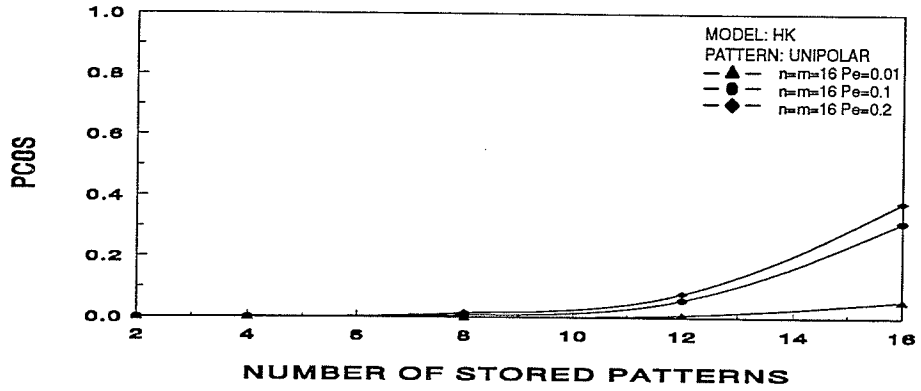

Fig. 3.23 The probability of HK model converges to oscillatory states. Memory configuration: n=16 and m=16. Noise: (the probability of each bit being reversed) P_e, is set to: 0.01, 0.1, and 0.2.

(4) *Spurious and oscillatory states*

Because the HK model utilizes nonsymmetric weights, the network can not guarantee that it will converge to stable states. However, the results presented in Fig. 3.23 indicate that oscillatory states rarely take place if the noise level (specified by P_e) is weak and the number of stored patterns, p, satisfies $p << \min(n, m)$. The probability of memory converging to oscillatory states (PCOS) is as low as 0.078 even when $P_e = 0.2$ and $p = 12$. Compared with spurious states, (depicted in Fig. 3.24) the problem of oscillatory states may be discounted if the number of stored patterns is small. Two other interesting phenomenons found in the testing are that (i) the oscillatory states become the major factor that deteriorates the memory performance if the number of stored pattens exceeds 12, and (ii) approximately 90% of spurious states in the HK model constitute false states. As a result, the HK model is not able to demonstrate its power in performing interpolative recall.

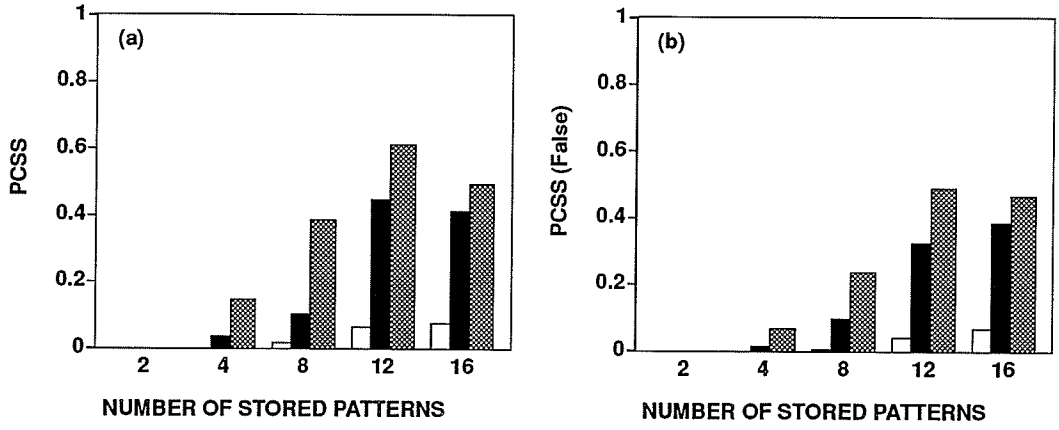

Fig. 3.24 (a): The probability of the HK model converging to spurious states. (b): The probability of the HK model converge to false spurious states. Memory configuration: n=16 and m=16. Input noise: the probability of each bit being reversed in input patterns, P_e is set to: □ P_e=0.01, ■ P_e=0.1, and ▩ P_e=0.2.

(4) *Learning iterations*

Although the Ho–Kashyap algorithm encodes information by optimal choice of a weight vector **w** and a margin vector **b** iteratively, the latter adjustable parameter **b** seems to be less important if training patterns are linearly independent. Test results show that if the number of stored associations $p << \min(n, m)$, the required number of iterations is approximately 1. This is because when $p << \min(n, m)$, randomly generated training patterns may be mutually linearly independent. Consequently, the error given in step (5) (in a Ho–Kashyap encoding algorithm) is brought to zero by one step through computing $\mathbf{w}^{*}_{(1)} = \mathbf{X}'^{+}\,\mathbf{b}_{(1)}$. This phenomenon reveals that the cost function for a linear independent training set may have a wide flat global minimum. The value of global minimum may not depend on the margin vector **b**. In the cases of $p << \min(n, m)$, the gradient $\frac{\partial E(\mathbf{w}, \mathbf{b})}{\partial \mathbf{w}} = 2\,\mathbf{X}^T(\mathbf{X}\,\mathbf{w} - \mathbf{b})$ in Eqn. (3.45) takes whole responsibility in finding solution weights. Experimental results show that

the number of iterations is larger than one only if the number of stored patterns, p, approaches or exceeds min(n, m). The number of iterations averaged over 400 trials for each tested model is listed in Table 3.5.

Table 3.5 Average Convergence Rate in the HK Encoding

Dimensions		The number of training pairs						
n	m	2	4	8	12	16	20	24
16	16	1.00	1.00	1.00	1.00	28.15	31.23	4423.64
16	8	1.00	1.00	2.16	22.58	79.25	103.74	2846.54
8	16	1.00	1.00	2.28	22.51	72.45	115.43	2958.07
8	8	1.00	1.00	2.28	50.66	N/A	N/A	N/A

Note: All data are averaged over 400 trials.

3. 5. 4 Summary

The HK model demonstrated relatively higher performance. The maximum capacity for the HK model configured as n=16, m=16 is approximately 23. Test results show that the capacity was bounded at the minimum dimension of an input or output pattern. This upper bound, however, can only be reached under the condition $P_e = 0$. The performance of correcting error in the HK model is superior to that in BAM especially when the number of patterns is larger than 4. For storing two associations, the ability to tolerate noise remains at the same level as that of BAM. The results also indicate that if the input pattern dimension is lower than the output pattern dimension, the effect of input and output pattern dimensions on accretive recall is not as significant as in BAM. The oscillatory states may be neglected if the number of stored patterns is small. It is necessary to point out that the higher performances of the HK model are achieved at the price of using nonsymmetric weights

which require a doubled space to store them. The other price that the HK model has to pay is the learning time. Compared to the Hopfield network and BAM (assumed as $n = m$), the HK model takes approximately $n^2p\ (8p-1) \times$ *the number of iterations* more encoding time.

3. 6. Backpropagation network for associative memory

Although the backpropagation algorithm (BP) [RuMc1986] is deemed as the most popular training paradigm and it has been widely investigated during the past few years, very little is known about the performance of the networks used as associative memories. What is the potential information capacity and error correction capability for this model whose weights are formed by the error adaptive correction algorithm? Is it superior to other models? Answers to these questions will be provided in this section.

3. 6. 1 Error backpropagation training algorithm

Backpropagation is a supervised learning rule for networks with hidden neurons. It is the generation of perceptron [MiPa1969] which is only comprised of input layer and output layer. For a multilayer network, the central problem is in obtaining the internal representation in hidden neurons. If the internal representation is known, weights associated with these neurons can be generated or adaptively modified by using the Hebbian or Ho–Kashyap encoding algorithm. The major practical limitation of perceptron approach is that the learning algorithm only can be applied to networks with a single layer of modifiable weights. This problem has been the bottle neck for quite a long time. The BP training algorithm [RuMc1986] successfully circumvents this problem by applying a chain of derivatives from the network output layer down to the input layer. The error information

collected at the output neurons is back propagated through the weights to form pseudo–errors for the neurons in the hidden layer. This allows hidden neurons to know what desired states are expected. Based on these pseudo–errors, weights associated with these hidden neurons are adjusted in a way that minimize these pseudo–errors. To give a specific example of this computing strategy, consider the network in Fig. 3.25.

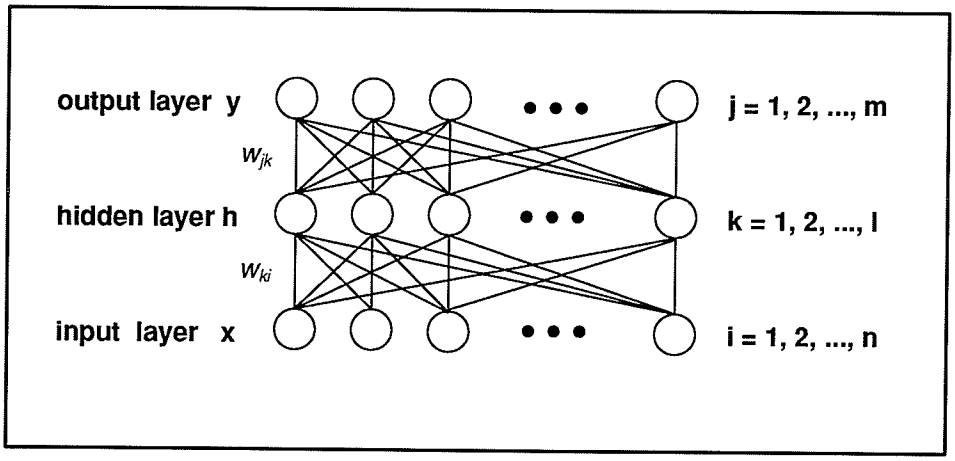

Fig. 3.25 The typical structure of multilayer perceptron.

The error function in the network output is defined as

$$E = \frac{1}{2} \sum_{s=1}^{p} \sum_{j=1}^{m} (y_j^{(s)} - y_j^{(d)})^2 \tag{3.47}$$

where $(y_j^{(s)} - y_j^{(d)})^2$ is the square error of element j between actual activation $y_j^{(s)}$ and the desired output $y_j^{(d)}$. During the training, this error is back propagated to the hidden layer. Weights are changed according to

$$\Delta w_{jk} = -\frac{\partial E}{\partial w_{jk}} = -\delta_j \, h_k \tag{3.48}$$

and

$$\Delta w_{ki} = -\frac{\partial E}{\partial w_{ki}} = -\zeta_k \, x_i \tag{3.49}$$

where

$$\delta_j = y_j \, (1 - y_j) \, (y_j^{(s)} - y_j^{(d)}) \tag{3.50}$$

and

$$\zeta_k = h_k \, (1 - h_k) \sum_{j=1}^{m} \delta_j \, w_{jk} \tag{3.51}$$

The derivation of Eqn. (3.48)–(3.51) is the chain rule starting from the output neuron to the input neuron. The generalized delta rule for adjusting the weights associated with the hidden neuron is formulated by

$$\Delta w_{jk}(t+1) = \eta_1 \, \Delta w_{jk}(t) - \alpha_1 \, \Delta w_{jk}(t-1) \tag{3.52}$$

and

$$\Delta w_{ki}(t+1) = \eta_2 \, \Delta w_{ki}(t) - \alpha_2 \, \Delta w_{ki}(t-1) \tag{3.53}$$

where t is the iteration number, $\Delta w_{jk}(t)$ and $\Delta w_{ki}(t)$ are calculated according to Eqn. (3.48) and (3.49) respectively; η_1 and η_2 are learning rates. The second term in Eqn. (3.52) and (3.53) is called a momentum term. It is introduced to increase the adaptive step if weight moves along a gently sloping floor and to decreases the adaptive step when it meets a sharp curvature which may lead to oscillation. Coefficients, α_1 and α_2, are constants which determine the degree of effect of past weight changes on the correct direction of movement in weight space [RuMc1986]. It is noted that the error information utilized for modifying weights is derived from the gradient, thus, a continuous differentiable nonlinear activation function is required.

Such a nonlinear activation function used in each neuron (except in input neurons) is in a sigmoid form:

$$F(x) = \frac{1}{1 + e^{-(x-\theta)}} \quad . \tag{3.54}$$

The trained network has the property that the error function defined in Eqn. (3.47) is minimized.

Although the BP algorithm circumvents many problems in the real world, using such an algorithm is usually frustrating. One of the major drawbacks of this method is its slow convergence rate [HiAn1989]. Starting from a random initial state, the path to the global minimum is often strewn with local minima [WeMa1991]. Another problem is the network architecture. Choosing the optimal number of hidden neurons and the number of hidden layers is not an easy task. Because the neurons in a hidden layer correspond to separated decision regions into which the training patterns are mapped [Lipp1987], too few hidden neurons cause networks to be unable to capture the essential features in training patterns, whereas too many hidden neurons may create the redundant hyperspace which causes the networks overreact in response to small (insignificant) changes in input patterns [Burr1986], [WeMa1991].

3. 6. 2 Training experiments

Because the performance of the BP network is significantly affected by the network topology, the task of performance evaluation is far more difficult. In order to meet the goal of comparability with other models as described in the previous sections, the topology for the BP network is carefully selected. The rule for determining if hidden neurons should be

introduced is based on whether networks are able to converge to a global minimum. Such a heuristic approach in determining the network structure begins with the network not containing hidden neurons. If the network fails to converge to a global minimum after twenty trials, hidden neurons with the same size of output arranged in one layer are added, otherwise, the structure remains unchanged.

All initial networks used in the testing contained only two layers: the input layer and the output layer. The dimensions of input and output layer were set to (n x m): 16x16, 16x8, 8x16, and 8x8. The first three models had the same dimension as *Model I*, *Model II*, and *Model III* used in testing the performance of BAM and the HK model. Each network was firstly trained by two and four associations; and then the number of training pairs was increased by four. Because the performance of the BP network depends on data representation, the training experiments were undertaken on four groups of training sets. Each set had a fixed number of associations p. These training associations were randomly generated under the *MD* constraint.

The network training was carried out by using PDP software [McRu1988]. The choice of learning mode was set to "set model fast 1" [Dole1991]. In this mode, the BP program updates its weights and thresholds according to the delta–bar–delta weight update rule [Robe1988], [Dole1991]. The default parameters mentioned in [Robe1988] were selected. All networks were trained by using a batch adapting mode (weight and threshold error derivatives were accumulated over an entire processing epoch and then the weights and the thresholds were modified) [McRu1988]. The error criteria were set to 0.01 for all models. The experimental results showed that the BP algorithm was easily to converge to a global minimum if the input dimension was higher than the output dimension. In the case of p = min(n, m), this was no guarantee for BP to find solution weights. The local minimum as well

as oscillatory problems occurred frequently. In this case, the training mode was changed to momentum, with a fixed learning rate of $\eta = 0.8$ and momentum coefficient $\alpha = 0.2$. These parameters were chosen because they provided a fast convergence rate and a high possibility in finding solution weights. The convergence rates averaging over 4 trials for each network's configuration are listed in Table 3.6.

Table 3.6 Average Convergence Rate in the BP Training

Dimensions		The number of training pairs			
n	*m*	2	4	8	16
16	16	20.25	39.22	96.34	202.03
16	8	18.33	45.65	63.87	162.65
8	16	30.12	112.24	236.50	9126.50*
8	8	25.34	64.34	145.34	N/A

Note: all data are averaged over 4 trials. Symbol ($*$) means the network contains hidden neurons with the same dimension as the output neuron.

3. 6. 3 Experimental results

The tests were mainly designed to investigate how the quality of associative recall in BP networks is affected by (i) the number of stored associations, (ii) noise in input patterns, and (iii) the spurious states. Because the BP network is a feed forward network the oscillation case need not to be considered. It should be emphasized that the results presented here are less reliable compared with those of the BAM and HK models. The inability to provide highly reliable results is due to the slow convergence rate in the training process. Another reason is that the standard PDP software does not provide automatic uphill movement. In a practical work this automatic uphill movement can be very helpful if a network is often

trapped in local minima. The need to manually control the learning rate and the momentum parameters is another problem that prohibits the training of large groups of associations within a limited time.

(1) *information capacity*

The capacity of the BP network was tested on three different models varied in their structures. The topological structures of tested networks were the same as those used in testing the performance of the BAM and HK models (*Model I model II* and *Model III*). However, there was one exception. This exception occurred in *Model II* (n=8, m=16, and p>8) where 16 hidden neurons arrange in one layer were introduced. The performance of information capacity is illustrated in Fig. 3.26. The result shows that (i) the BP network is

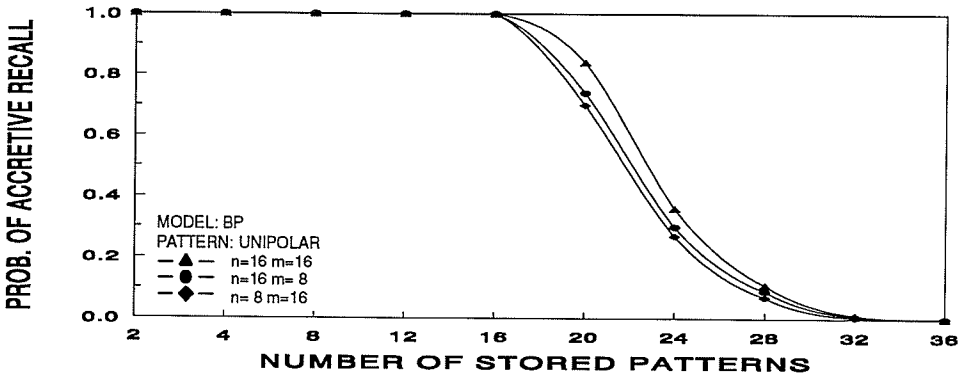

Fig. 3.26 Information capacity of the BP network (accretive recall). *Model I* (n=16, m=16.), *Model II* (n=16, m=8) and *Model III* (n=8, m=16). Pe, the probability of each bit being reversed in input patterns, equals 0.

guaranteed to recall all training patterns if the number of stored associations, p, satisfies $min(n+1, m+1)$, and (ii) for three tested BP networks, the capacity gaps in terms of accretive

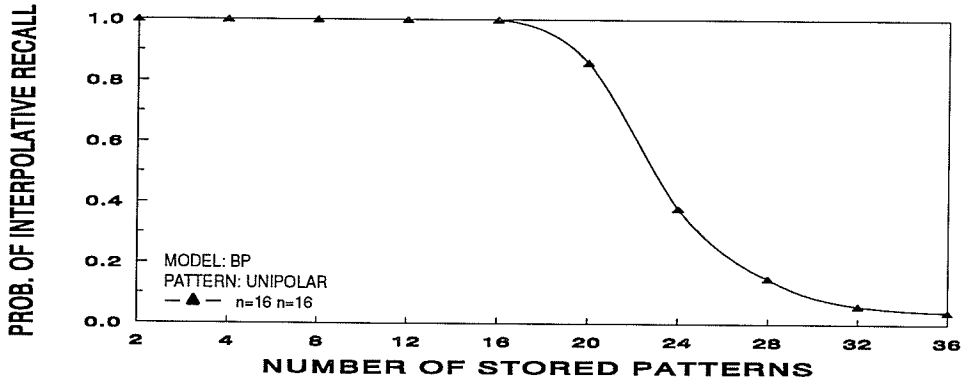

Fig. 3.27 Information capacity of the BP network (interpolative recall). *Model I*. Memory configuration: *n=16*, *m*=16. *Pe*, the probability of each bit being reversed in input patterns, equals 0.

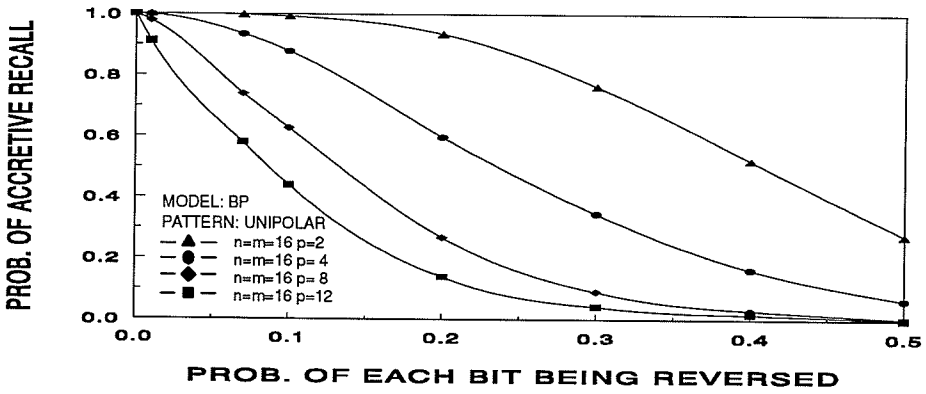

Fig. 3.28 The error correction capability of the BP network (accretive recall). Memory configuration: *n*=16 and *m*=16. Memory load: $p = 2, 4, 8$, and 12.

recall are not obvious. Fig 3.27 illustrates the performance of interpolative recall of *model I*. The capacity for this model is about 17. It remains at the same level as that of accretive recall.

(2) *error correction capability*

The results of error correction for BP networks is presented in Fig. 3.28 and 3.29. The results indicate that although the capacity of the BP network is substantially higher than the Hopfield network and BAM, the maximum noise level (denoted by P_e) that the BP network is able to tolerate approximately equals 0.2 (tested under the condition of $p = 2$). However, the situation is changed as the number of stored associations increases. The BP network outperforms both the Hopfield and BAM models if the number of stored patterns/associations is larger than 2. In terms of accretive recall, the performance remained at the same level compared to that of the HK model.

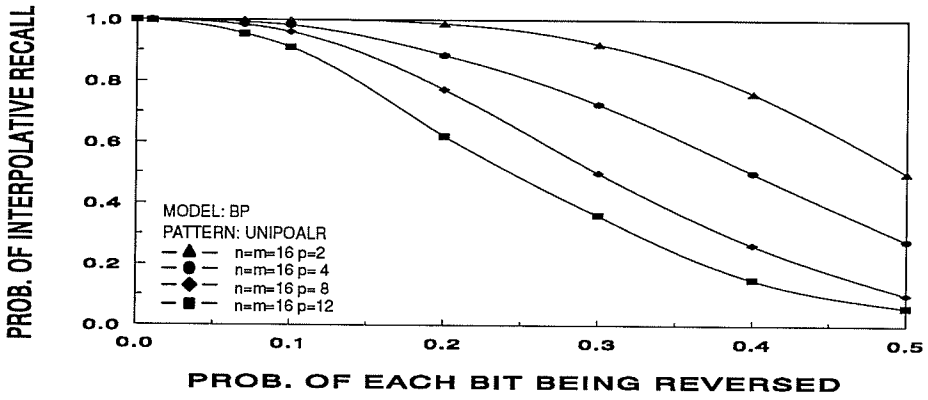

Fig. 3.29 The error correction capability of the BP network (interpolative recall). Memory configuration: n=16 and m=16. Memory load: p = 2, 4, 8, and 12.

(3) *The effect of input and output pattern dimensions on accretive recall*

In order to measure the effect of input and output pattern dimensions on accretive recall, an additional BP network, which was configured as n=8, m=8, was introduced. The measurement of this effect was based on the performance statistics collected during the

testing. The effect of input and output pattern dimensions on accretive recall was calculated according to the Eqn. (2.7) defined in section 2.2.3 in chapter 2. To meet the requirement of comparability with the results from other tests, similar test points were selected, i.e., P_e =0.07 and 0.2. The result of the effect is illustrated in Fig. 3.30.

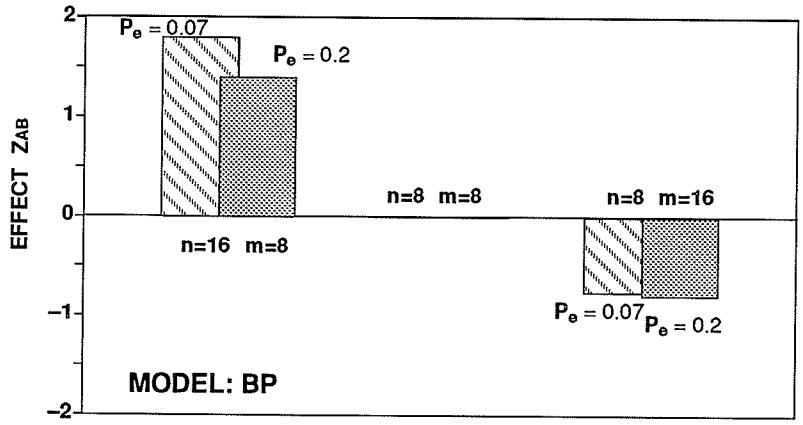

Fig. 3.30 The effect of input and output patterns dimensions on accretive recall. Positive value suggests the increase in the performance, and the negative value denotes the decrease in the performance. Here n and m denote the dimensions of input and output patterns respectively. P_e stands for the probability of each bit being reversed in input patterns.

(4) *Spurious states*

The performance of accretive recall shows that the trained BP network is guaranteed to recall all stored patterns, namely, if the input pattern is not contaminated with noise, the probability for BP network mapping into spurious memory will be zero. Furthermore, the high capacity also prevents from the BP network mapping into spurious states. Fig. 3.31 depicts the results of PCSS testing. The tested model is configured as n=16 m=16. The most interesting phenomenon found in this testing is that although there are a large number of spurious states, quite a few of them belong to the category of false states. This phenomenon

helps to explain the reason why the BP network is able to achieve a high probability of interpolative recall.

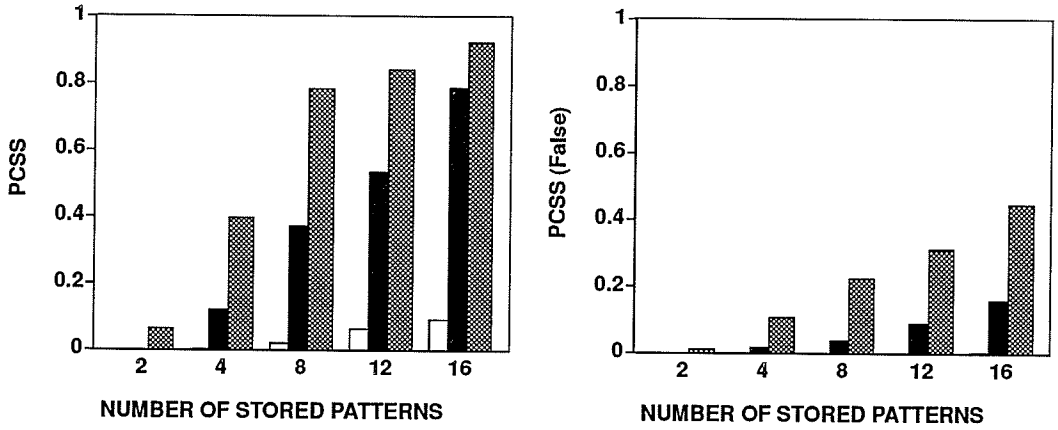

Fig. 3.31 (a): The probability of the BP network converging to spurious states. (b): The probability of the BP network converge to false spurious states. Memory configuration: n=16 and m=16. Input noise: the probability of each bit being reversed in input patterns, P_e is set to: □ P_e =0.01, ■ P_e =0.1, and ▩ P_e =0.2.

3. 6. 4 Summary

The performances of the two–layer and three–layer feed forward BP network used for associative memories were investigated. The investigation was undertaken on 12 different BP networks varied with their input and output dimensions. Those tested models had the same structure as BAM and HK models except for *model III* (n=8, m=16, and p=16) which contained 16 hidden neurons arranged in one hidden layer. The results of the experiments show that if $p > \min(n, m)$ the BP network fails to converge (sum–square–error is larger than 1) unless hidden neurons are introduced (the number of layers and the number of neurons need to be added into the network depending on the number of pairs of training patterns and the mutual coupling among training patterns). The results also show that the trained

networks are guaranteed to recall all stored associations if the network is evoked by noiseless inputs. The effect of input and output pattern dimensions on accretive recall is relatively significant compared with both BAM and HK models. The probability for the BP network mapping into spurious states (false) was significantly lower than other models.

3. 7. References

[Burr1986] Burr, D. (1986), "A neural network digit recognizer", Preceding of the IEEE Conference on System, Man and Cybernetics, pp. 1621–1625.

[Dole1990] Dolenko, B. (1990), "Backpropagation neural net simulator, Reference Manual", Department of Electrical and Computer Engineering, University of Manitoba.

[HaHe1988] Haines, K. R. and Hecht–Nielsen, R. (1988), "BAM with increased information storage capacity", Proceedings, International Conference on Neural Networks, San Diego, CA: SOS Print.

[Hass1989] Hassoun, M. H. (1989), "Dynamic heteroassociative memories", Neural Networks, Vol. 2, pp. 275–287.

[HaYo1989] Hassoun, M. H. and Youssef, A. M. (1989), "High performance recording algorithm for Hopfield network", Optical Engineering, Vol. 27, No.1, pp. 46–54.

[Hebb1960] Hebb, D. O. (1960), "The organization of behavior", New York: Wiley.

[HeKP1991] Hertz, J., Krogh, A. and Palner, G. R. (1991), "Introduction to the Theory of Neural Computation", Addison–Wesley, Redwood City, CA.

[HiAn1989] Hinton, G. E. and Anderson, J. A. (Eds.) (1989), "Parallel Models of Associative Memory", Updated Edition, Hillsdale, N. J. : Lawrence Erlbaum Associates.

[HoKa1965] Ho, Y. and Kashyap, R. L. (1965), "An algorithm for linear inequalities and its applications", IEEE Transactions on Electronic Computers, Vol. 14, pp. 683–688.

[Hopf1982] Hopfield, J. J. (1982), "Neural networks and physical systems with emergent collective computational abilities", Proceedings of the National Academy of Science USA, Vol. 79, pp. 2554–2558.

[Hopf1984] Hopfield, J. J. (1984), "Neurons with graded response have collective computational properties like those of two–state neurons", Proceeding of the National Academy of Sciences, U.S.A., Vol. 81, PP. 3088–3092.

[Jeru1984] Jeruchim, M. C. (1984), "Techniques for estimating the bit error rate in the simulation of digital communication systems", IEEE Journal Selected Areas in Communication, SAC–2, pp. 153–170.

[Kosk1987] Kosko, B. (1988), "Adaptive bidirectional associative memories", Applied Optics, Vol. 26, pp. 4947–4960.

[Kosk1988] Kosko, B. (1988), "Bidirectional associative memories", IEEE Transactions on System, Man and Cybernetics, Vol. 18, No. 1, pp. 49–60.

[Lipp1987] Lippmann, R . P. (1987), "An introduction to computing with neural nets", IEEE ASSP Magazine, April, pp. 4–22.

[MaPa1969] Minsky, M. and Papert, S. (1969), "Perceptrons", The Massachusetts Institute of Technology, Printed by Maple Press Company.

[McRu1988] McCelland, J. L. and Rumelhart, D. E. (1988), "Exportation in Parallel Distributed Processing: a handbook of models, programs, and exercises", The MIT Press, Cambridge, Massachusetts.

[MPRV1987]McEliece, R. J., Posner, E. C., Rodemich, E. R. and Venkatesh, S. S. (1987), "The capacity of the hopfield associative memory", IEEE Transactions on information Theory, Vol. 33, pp. 461–482.

[Robe1988] Robert, A. J. (1988), "Increased rates of convergence through learning rate adaptation", Neural Networks, Vol. 1, No. 4, pp 295–307.

[Ross1987] Ross, S. M. (1987), "Introduction to Probability and Statistics for Engineers and Scientists", John Wiley & Sons.

[RuMc1986] Rumelhart, D. E., McCelland, J. L and the PDP Research Group, (1986), "Parallel Distribution Processing: Exploration in the Microstructure of Cognition, Vol. 1, Foundations", The MIT Press Cambridge, Massachusetts.

[SkWa1981] Sklansky, J. and Wassel, G. N. (1981), "Pattern Classifiers and Trainable Machines", Springer–Verlag, New York Inc..

[WeMa1991] Weymaere, N. and Martens, J.–P. (1991), "A fast robust algorithm for feedforward neural networks", Neural Networks, Vol. 4, pp. 361–369.

CHAPTER 4

COMPARISONS

4. 1. Introduction

The preceding chapter has shown how the proposed testing procedure was applied to investigating four different models of associative memories. In this chapter, an attempt is made to compare their performances. The discussions as well as additional findings from the comparison will also be presented. Because the performance gap between certain types of models may not be wide for some properties, the histogram chart plotted at each test point is adopted in order to make these gaps more obvious. The comparisons focus on

- information capacity,
- the ability to correct noise in input patterns,
- the effect of input and output pattern dimensions on accretive recall,
- spurious states, and
- the computational complexities in encoding and associative recall.

4. 2. Information Capacity

Experimental results presented in chapter 3 have shown that the difference of capacity between any two tested models become appreciable if the number of stored patterns/associations is larger than two. For this reason, the comparison begins with storing

four patterns/associations and than increasing the load by four. The capacity comparison taken on four different memory models is divided into two major categories. One is concerned with the quality of recall stored patterns/associations perfectly (accretive recall); the other is the performance of interpolative recall. It will be seen that these two different types of recall lead to two different ways in appraising the performance of associative memory. In this section, the performance comparison will be extended to the situation that the memory input is corrupted with noise. Although this situation is beyond the scope of analyzing the memory capacity, some properties behind the test condition that the memory is evoked by training patterns will be discovered. This situation is somewhat different from the error correction since this comparison mainly focuses on how the quality of associative recall is affected by different memory loads.

The first comparison was taken on the accretive recall. Plotting all results from four tested models as well as taking into account the noise in input patterns made comparison much easier. From Table 4. 1 and Fig. 4.1, one may immediately deduce that the HK model

Table 4. 1 Summary of Information Capacity

Model	Criterion	Accretive	Interpolative
HOP	$P_r = 0.95$	3	4
BAM	$P_r = 0.95$	3	4
HK	$P_r = 0.95$	23	23
BP	$P_r = 0.95$	17	17

Memory configuration: Hopfield network: n=16, BAM, HK, and BP: n=16, m=16. Test condition: no noise in input patterns.

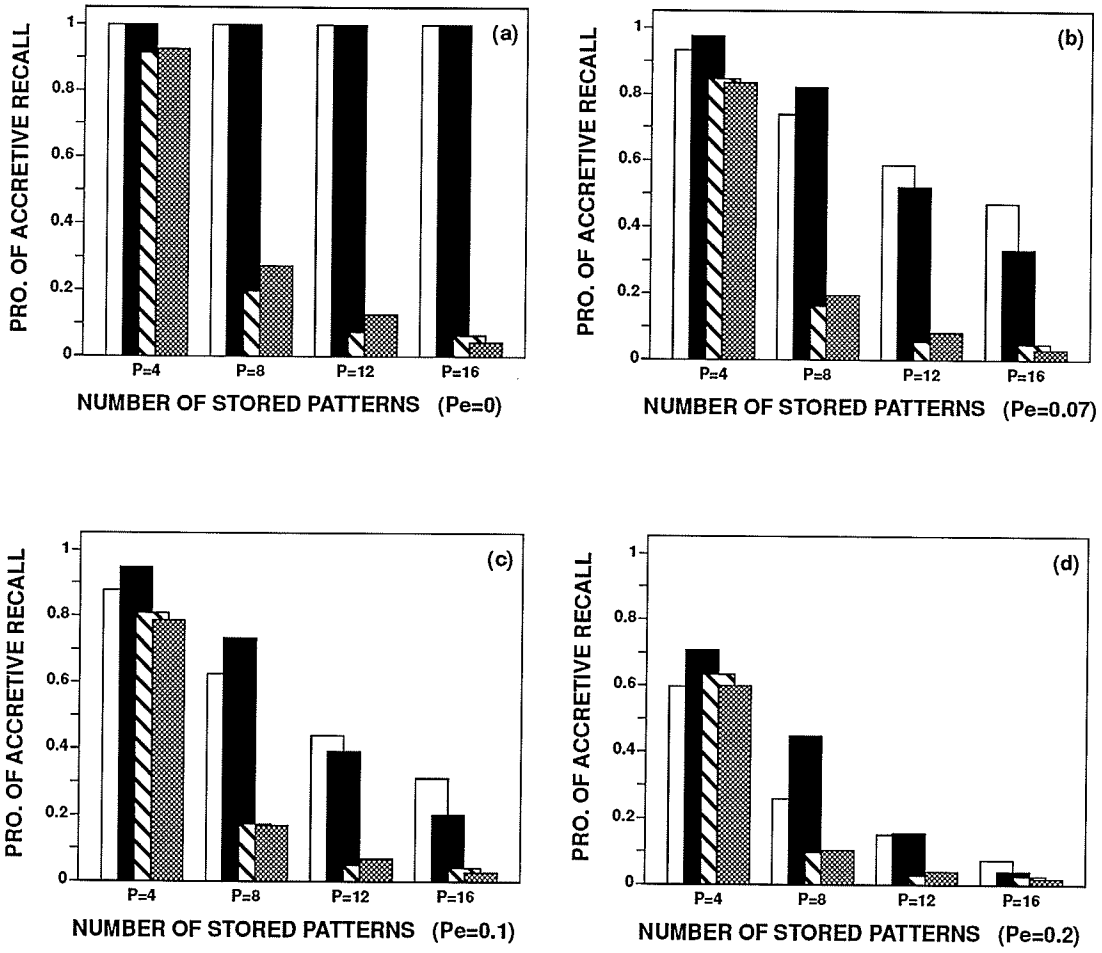

Fig. 4.1 Performance comparison: information capacity (accretive recall). Tested models are configured as: (i) BP, HK, BAM models: n=16, m=16, (ii) Hopfield network: n=16. ☐ BP network, ■ HK model, ▨ BAM, ▩ Hopfield network.

and BP network are able to achieve higher probability of accretive recall which leads to the higher information capacity. The Hopfield networks and BAM never perform competitively. Furthermore, the HK model appears to be able to achieve higher quality of accretive recall than BP if p is small (see Fig. 4.1b and 4.1c). However as p increases, the degradation in performance in the HK model is more significant than in the BP network.

One property which may not be immediately apparent in Fig. 4.1 is that BAM seems to have more difficult in performing accretive recall than the Hopfield network if the noise level in the input pattern is low. The essence behind this superficial phenomenon is that BAM can only recall re–stored association if and only if this association is a local minimum in an energy surface [Kosk1987], [Kosk1988]. The problem, however, lies in the fact that the encoding algorithm proposed by Kosko can not guarantee that any stored association is a local minimum. This issue has been investigated by Y.–F. Wang in his recently published paper [WaGM1991]. Another reason which results in BAM being unable to recall stored association is the reversed version of training patterns as well as the bidirectional search process. Because the fundamental Hebbian rule can not guarantee that the energy of any training association is a local minimum, the bidirectional search may not always help BAM to converge to a correct state. This hypothesis has been proved experimentally in this work. The result is presented in *Appendix I*. This result manifests that the bidirectional recall is sometimes detrimental for the quality of associative recall.

Of the interpolative recall, the BP network constantly performs best. The HK model is unable to compete with the BP network if the input pattern is corrupted with noise (see Fig. 4.2). From these results, it can be conjectured that the HK model may be best–suited to perform accretive recall especially when a few associations are stored, while the BP network may win favor in the area where interpolative recall is essential.

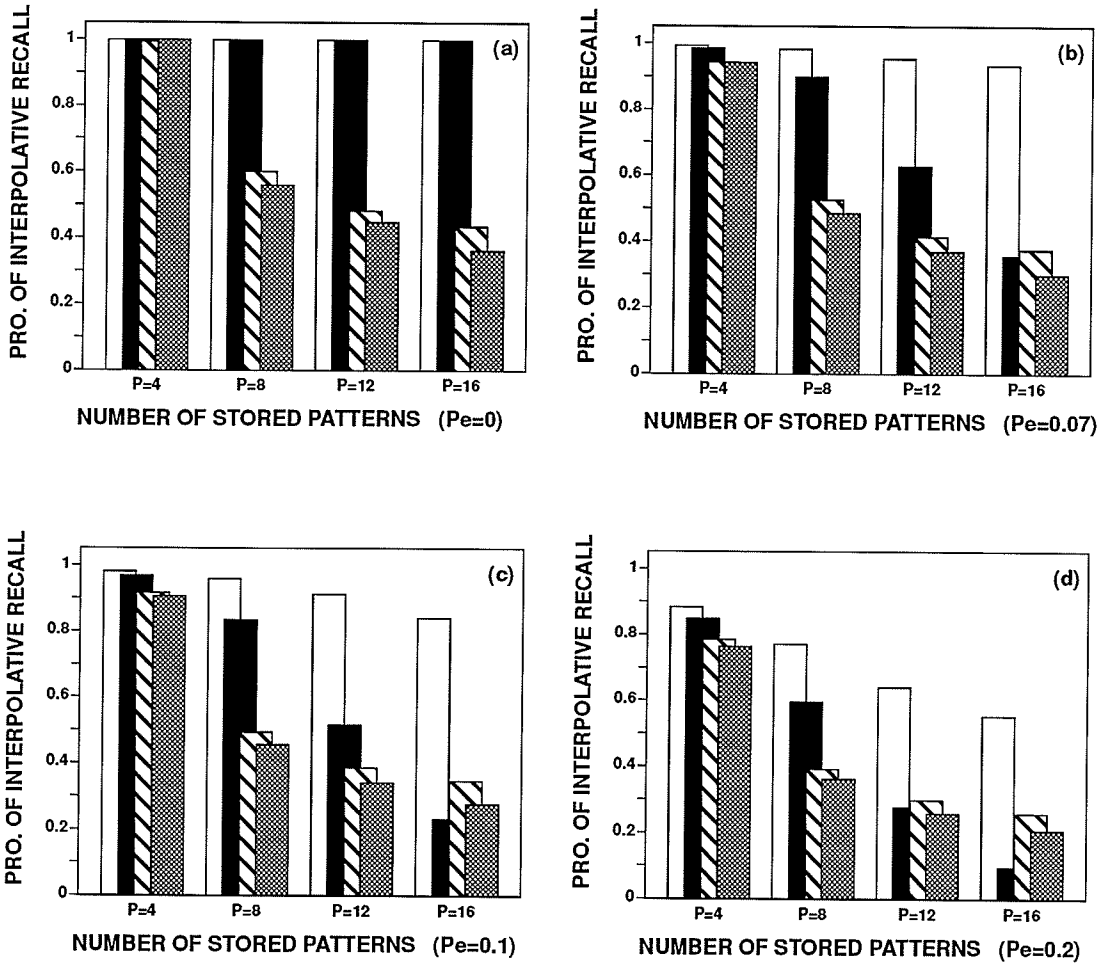

Fig. 4.2 Performance comparison: information capacity (interpolative recall). Tested models are configured as: (i) BP, HK, BAM models: n=16, m=16, (ii) Hopfield network: n=16. ☐ BP network, ■ HK model, ▨ BAM, ▩ Hopfield network.

4. 3. The Ability to Tolerate Noise

Similar investigation was carried out in order to capture the performance of error correction capability. As before, the HK model demonstrates superiority in accretive recall if the number of stored patterns is less than eight; while the BP network demonstrates the

best performance in interpolative recall. These results are shown in Fig. 4.3 and Fig. 4.4. Some intrinsic behavior captured in the investigation is that (i) the relative performance of the Hopfield networks and BAM are approximately at the same level in the error correction

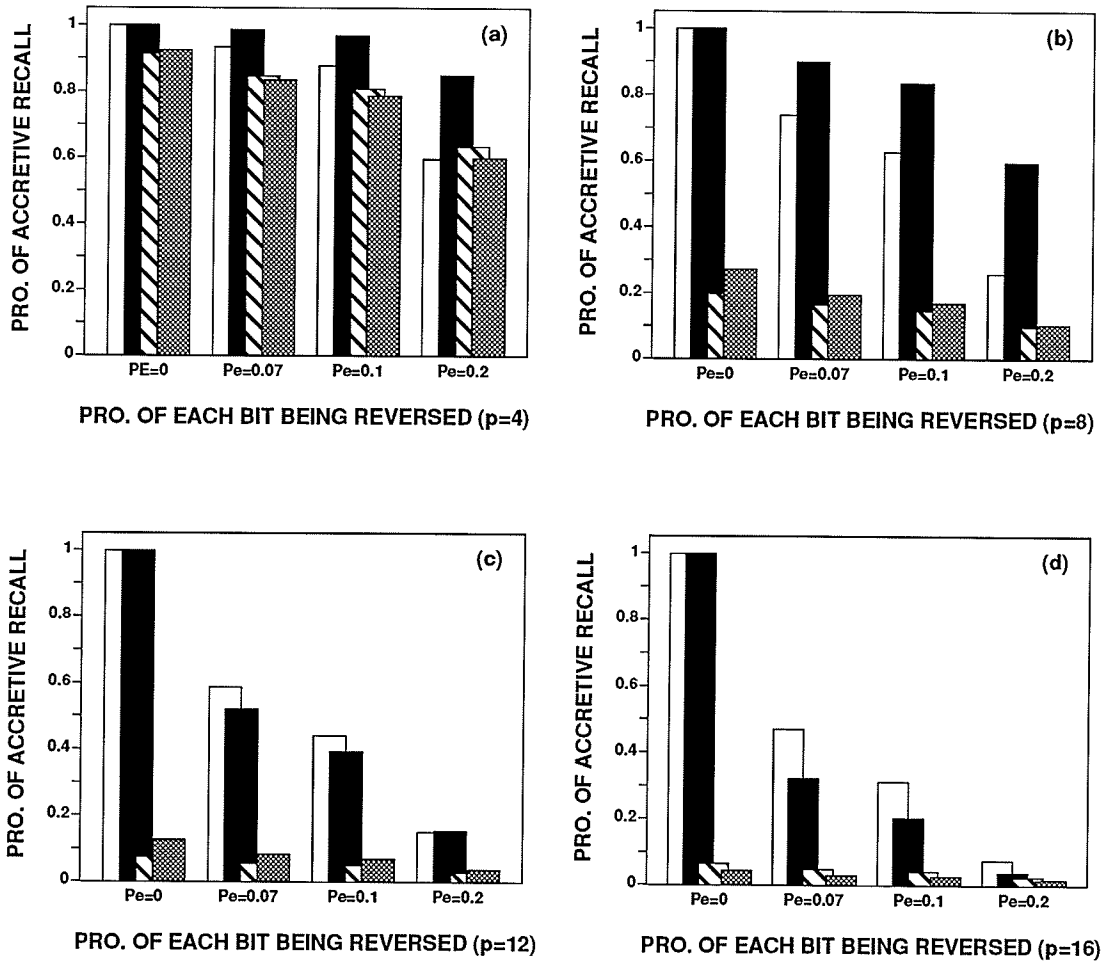

Fig. 4.3 Performance comparison: error correction capability (accretive recall). Tested models are configured as: (i) BP, HK, BAM models: n=16, m=16, (ii) Hopfield network: n=16. ☐ BP network, ■ HK model, ▨ BAM, ▩ Hopfield network.

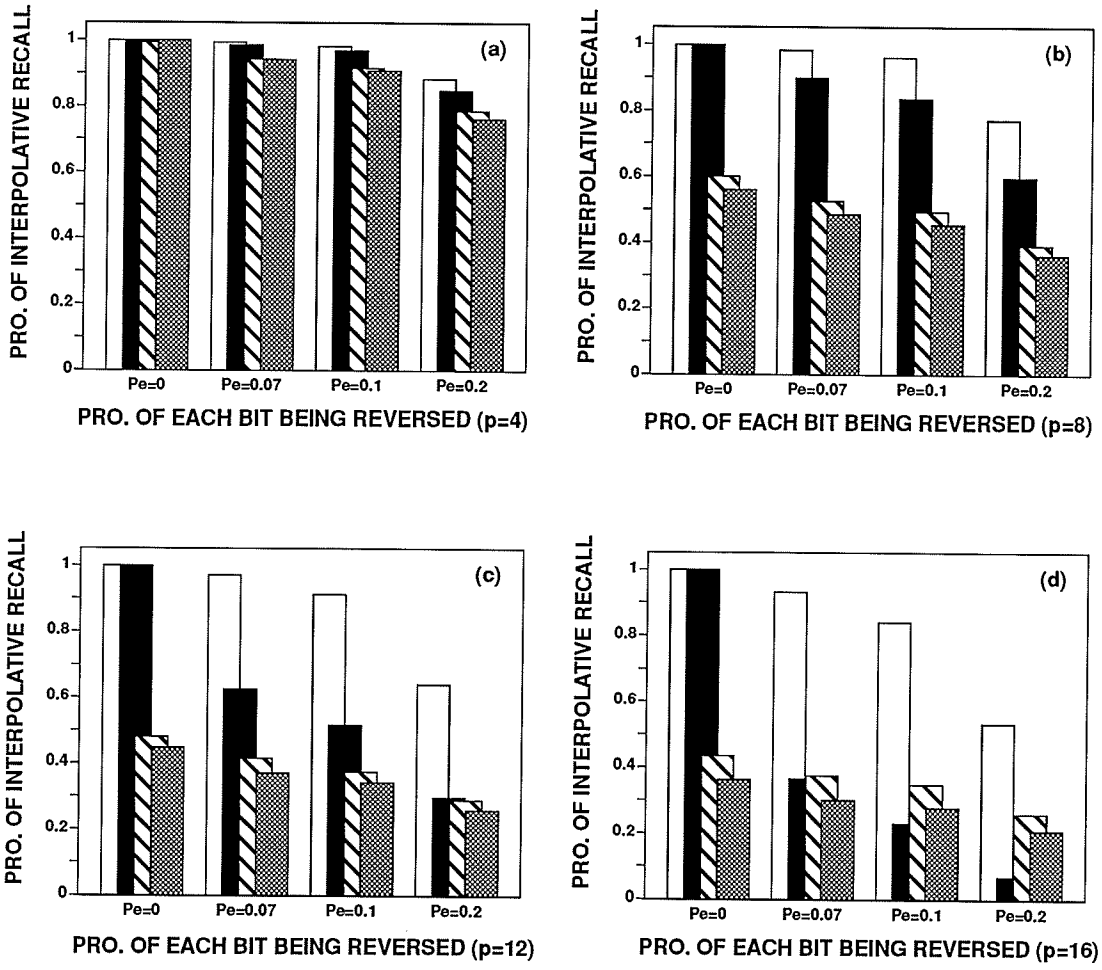

Fig. 4.4 Performance comparison: error correction capability (interpolative recall). Tested models are configured as: (i) BP, HK, BAM models: n=16, m=16, (ii) Hopfield network: n=16. □ BP network, ■ HK model, ▨ BAM, ▩ Hopfield network.

aspect, (ii) none of the models are able to perform an acceptable level of associative recall if noise level, P_e, is higher than 0.2 (tested under the condition of storing more than four patterns/associations), and (iii) for both accretive recall and interpolative recall the performance of the HK model appears to be approximately identical. This phenomenon implies that the bidirectional HK model behaves less "flexibly" than other models. It either successfully converges to correct states or slips to false states.

4. 4. The Effect of Input and Output Pattern Dimensions on Accretive Recall

Experimental results provided in Chapter 3 have verified that the quality of accretive recall is somewhat affected by the input and output pattern dimensions. In view of the results, one may conjecture that improvement in the performance of accretive recall is achievable if the input pattern dimension is higher than that of the output pattern dimension ($n > m$), and the performance decreases if the input pattern dimension is lower than the output pattern dimension ($n < m$). It ought to be noticed that the effect of input and output pattern dimensions on accretive recall is measured under the condition that random errors occur in the input patterns. In general, the selectivity of associative mapping is better if the input patterns are further distant from each other.

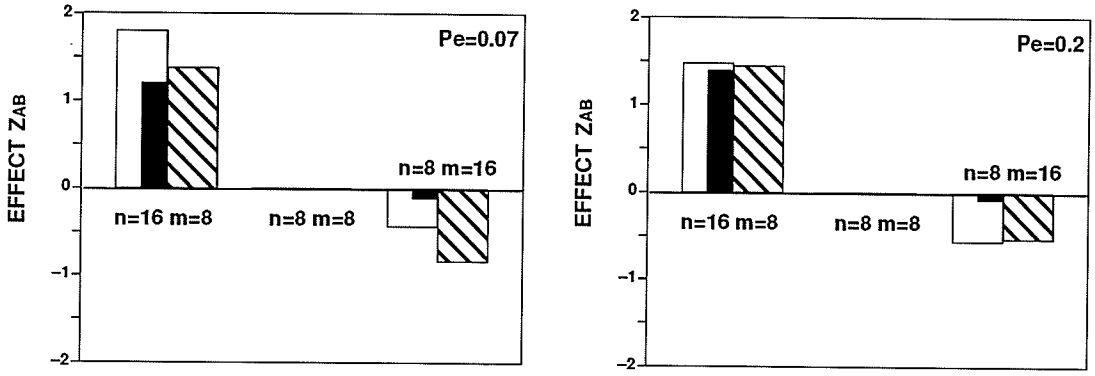

Fig. 4.5 Performance comparison: the effect of input and output pattern dimensions on information capacity (accretive recall). Tested models are configured as: (i) BP, HK, BAM models: n=16, m=8, n=8, m=8, and n=8, m=16. ☐ BP network, ■ HK model, ▨ BAM. The positive values indicate the increase in the performance, whereas the negative values stand for the decreases in the performance.

The experimental results also demonstrate that the degree of the effect of input and output pattern dimension on accretive recall is encoding algorithm dependent. These effects measured by means of simulation of BAM, HK and BP memories are depicted in Fig. 4.5. It should be pointed out that throughout the comparison, the performance of the memory model configured as n=8 and m=8 provided a baseline reference. The results given in the Fig. 4.5 show that the HK model exhibits the least effect, especially when $n < m$.

4. 5. The Spurious States

This section is only concerned with the comparison of the probability of tested memories converging to spurious states (false) during the associative recall. The discussion of another type of false states, oscillatory states, is omitted here because they were only detected in the HK model. The probability of associative memories converging to (false) spurious states is depicted in Fig. 4.6. These figures suggest that both Hopfield and BAM suffer a serious spurious state problem. It has been shown [HaYo1989] that these spurious states are composed of two different classes: one is, of course, the reversed version of training patterns, the other is the Boolean combination of training patterns. Substantial differences can be found by comparing the performance of Hopfield and BAM models with HK and BP networks. The unidirectional BP network demonstrates the lowest PCSS for most cases. The lowest PCSS may be contributed to the BP training algorithm which is capable of generating highly precise weights and thresholds through minimizing the sum–of–square–error between actual responses and the desired outputs iteratively. The HK

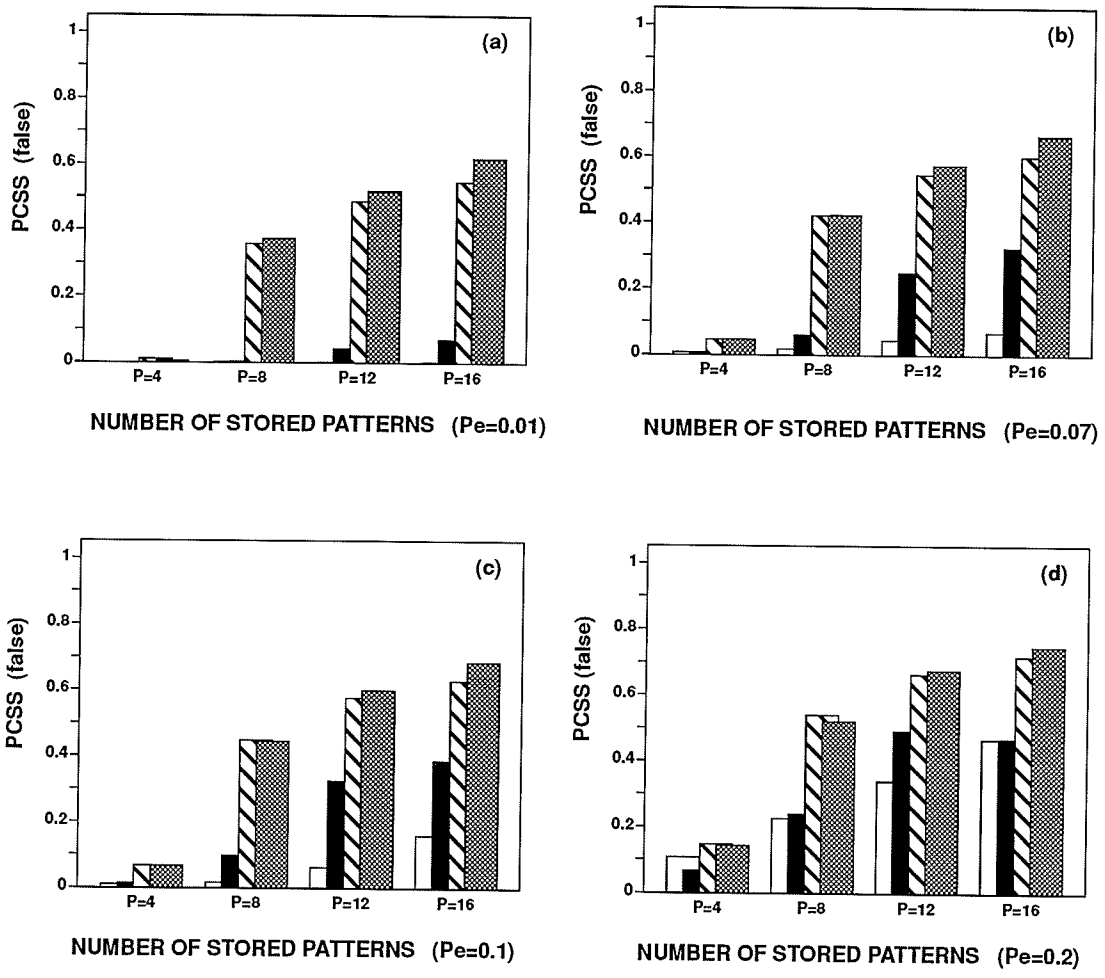

Fig. 4.6 Performance comparison: the probability of memory converging to false spurious states, PCSS (false). Tested models are configured as: (i) BP, HK, BAM models: n=16, m=16, (ii) Hopfield network: n=16. □ BP network, ■ HK model, ▨ BAM, ▩ Hopfield network.

algorithm, however, can only produce least–square solution and provide less flexible connecting weights. Because the Hebbian rule is unable to generate the weights and thresholds with the same dynamic range as those generated by Ho–Kashyap and BP algorithms (see appendix II), the performances of Hopfield and BAM models, therefore, can not compare with those of the HK and BP networks.

In conjunction with the high performance in interpolative recall, it is concluded that for the BP network, there must exist a large number of (nonfalse) spurious states located around the stored training pairs. These states improve the quality of interpolative recall. The HK model, however, tends either to converge to true states or to slip to the false states.

4. 6. The Analysis of Computational Complexity in Encoding and Recall

Perhaps the most difficult work is to compare the computational requirement. This difficulty is attributed to the fact that both HK and BP encoding algorithms include steps which are executed only when certain conditions are met by the data at the given iteration. This phenomenon implies that the actual computation time on every iteration is not a constant. Another difficulty is due to the fact that the computation requirement is sensitive to the training pattern that is selected, the learning parameters that are used, and the error criteria (for BP) which is chosen. For these reasons, the computational complexity measure is not meaningful unless all optional conditions are specified and statistical analysis is used. Even though one is able to meet the conditions which are described above, the results may still not be reliable because the BP algorithm frequently suffers from local minimum problem, especially as p approaches the dimension of min(n, m) or hidden neurons are introduced. For all the reasons addressed above, the results presented in Table 4. 2 should be taken to be no more than an approximate guide to the relative complexities except those of the Hopfield and BAM models.

(1) *Temporal complexity of encoding algorithm*

In the measure of the computational complexity, a cost function, $O(c_1n, c_2m, c_3p)$, is adopted, where c_1, c_2, and c_3 are constants, n and m refer to input and output pattern dimensions and p denotes the number of patterns/associations stored in a memory. This

computer independent measure is used to express the order of the time required to execute the algorithm on the problem size n and m and p. Accordingly, if the algorithm is said to execute $O(2n^2m)$ time, it is equivalent to be expressed as follows: the amount of work required by the algorithm is proportional to $2n^2m$ [Kron1987]. Table 4.1 presents the results of the computational complexity measure of the algorithms used in implementating Hopfield, BAM, and HK models of associative memories. The complexity analysis of the BP algorithm is exclusive because the computational requirement is extremely sensitive to many conditions. It should be pointed out that the number of iterations has not been taken into account in the analysis of computational complexities. For this reason, the results shown in Table 4. 2 can only be treated as a unit complexity. The actual computational requirement for the Ho–Kashyap encoding algorithm is the number of iterations (see Chapter 3, section 3.73) times the unit complexity.

Table 4. 2 Computational Complexity

Model	Complexity
HOP	$O(n^2p)$
BAM	$O(nmp)$
HK	$O(4n^2p^2 + 4m^2p^2)$

(2) *Temporal complexity of associative recall*

For a dynamic associative memory, the computational requirement depends on both the size of the memory and the number of iterations. A unit execution time (one iteration) will be a constant if the size of memory is fixed. For this reason, the analysis of computational complexity can be focused on the number of iterations that is required in an associative recall. Tables 4. 3–4. 5 give the averaging results of the Hopfield network, BAM, and the HK model. Since the number of iterations is affected by both the noise level in the input

pattern and the number of stored patterns in the memory, the parameters, P_e and p, are specified.

Table 4. 3 Average Number of Recall Iterations (Hopfield)

p	$P_{e=0.01}$	$P_{e=0.07}$	$P_{e=0.1}$	$P_{e=0.2}$
4	1.15	1.69	1.81	1.97
8	1.22	1.76	1.90	2.14
12	1.23	2.48	2.56	2.72
16	1.52	2.69	2.72	2.83

Table 4. 4 Average Number of Recall Iterations (BAM)

p	$P_{e=0.01}$	$P_{e=0.07}$	$P_{e=0.1}$	$P_{e=0.2}$
4	1.15	1.68	1.81	1.97
8	1.71	1.71	1.85	2.95
12	2.25	2.46	2.53	2.70
16	2.61	2.68	2.77	2.88

Table 4. 5 Average Number of Recall Iterations (HK)

p	$P_{e=0.01}$	$P_{e=0.07}$	$P_{e=0.1}$	$P_{e=0.2}$
4	1.15	1.69	1.82	1.97
8	1.53	1.69	2.04	2.04
12	1.38	1.82	3.45	4.24
16	1.95	5.42	6.27	7.29

4. 7. References

[Hass1989] Hassoun, M. H. (1989), "Dynamic heteroassociative memories", Neural Networks, Vol. 2, pp. 275–287.

[HaYo1989] Hassoun, M. H. and Youssef, A. M. (1989), "High performance recording algorithm for Hopfield network", Optical Engineering, Vol. 27, No.1, pp. 46–54.

[Koho1984] Kohonen, T. (1984), "Self–organization and Associative Memory", Berlin: Springer–Verlag.

[Kosk1987] Kosko, B. (1987), "Adaptive bidirectional associative memories", Applied Optics, Vol. 26, pp. 4947–4960.

[Kosk1988] Kosko, B. (1988), "Bidirectional associative memories", IEEE Transactions on System, Man and Cybernetics, Vol. 18, No. 1, pp. 49–60.

[Kron1987] Kronsjo, L. (1987), "Algorithm: the Complexity and Efficiency", Second Edition. John Wiley & Sons.

[WaGM1991]Wang, Y–F., Cruz, J. B. and Mulligan, J. H. (1991), "Guaranteed recall of all training pairs for bidirectional associative memory", IEEE Transactions on Neural Networks. Vol. 2, No. 6, pp. 559–567.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5. 1. Conclusions

The testing procedure that was developed for this study have been used to investigate the performance of associative memory. The procedure follows the black box test strategy. Although such a test strategy is based on the probabilistic and statistical analysis, reliable test results were obtained with low computation cost. The high reliability and the low computational requirement contribute to the minimum distance approach proposed in this thesis. This approach also allows for estimating the extreme–performance characteristics of associative memories.

The performance investigation has focused on four major properties: information capacity, error correction capability, the effect of input and output dimensions on accretive recall, and the probability of memory converging to false states. The computational complexity of encoding and associative recall were also investigated. Conclusions drawn from this investigation are as follows:

The error correction performance of associative memories based on the Hebbian learning rule, such as the Hopfield network and BAM, is affected by the reversed version of the training patterns unavoidably stored in the memory during the encoding. However, if the reversed version of training patterns are permitted to represent the same information as the original patterns, improvement in interpolative recall can be obtained, especially when a few patterns/associations are stored in the memory (see Appendix III).

The BAM's capacity is much lower than Kosko's original estimation, min(n, m). The actual capacity can not exceed $n/2\log_2 n$, which was analytically derived by McEliece, et al. to estimate the asymptotic capacity of a Hopfield network with n neurons [MPRV1987]. This value can be directly applied to estimating the capacity of the bidirectional associative memory.

Experimental results reveal that the bidirectional search strategy used in BAM and HK models does not provide significant improvement in accretive recall. It sometimes deteriorates the performance of interpolative recall.

The Ho–Kashyap encoding algorithm provides a significant improvement in accretive recall. The maximum associations that can be stored in the HK model are approximately min(n, m) if $n \neq m$. However, if the memory is constructed as $n = m$, the capacity is able to reach approximately $1.4n$ (tested at n=m=8 and n=m=16).

The HK encoding algorithm allows for the highest performance in accretive recall if the number of stored pattern p is small with respect to the input and output pattern dimensions.

The memory implemented by the backpropagation algorithm is guaranteed to recall all stored training pairs if the memory is evoked by noiseless inputs.

The BP network seems to be best–suited to the situation where interpolative recall is more important than accretive recall. The superiority in this respect becomes more pronounced as many associations are stored.

All heteroassociative memory show improvement in performance when $n > m$, and degradation in performance when $n < m$. Among these, the HK model is least affected by the dimension of input and output patterns.

Of all tested models, the HK model demonstrates the lowest probability of converging to spurious states. In contrast, the BP network shows the highest probability in this respect.

For HK model, 95% spurious states constitute false states. For the BP network, however, only 35% spurious states belong to the false states category.

For the Ho–Kashyap encoding algorithm, the average number of iterations required for encoding is approximately 1 if the number of stored patterns is much less than min(n, m).

The summarized investigation results are shown in Table 5.1.

Table 5.1 Summary of Investigation Results

Memory Model	Capacity (accretive)	Capacity (interpolative)	Err–corre (accretive)	Err–corre (interpolative)	Spurious states	Spurious states (false)	oscillatory states	System complexity
Hopfield Network	3	4	Pe≈0.02	Pe≈0.10	Pr=0.93	Pr=0.61	Pr=0.00	Weights: n x n Encoding: O(n^2p) Recall: less then 3 iterations
BAM	3	4	Pe≈0.02	Pe≈0.12	Pr=0.91	Pr=0.54	Pr=0.00	Weights: n x m Encoding: O(nmp) Recall: less then 3 iterations
HK Model	23	23	Pe≈0.12	Pe≈0.15	Pr=0.00	Pr=0.00	Pr=0.00	Weights: n x m Encoding: O($4n^2p^2 + 4m^2p^2$) Recall: less then 8 iterations
BP Network	17	17	Pe≈0.08	Pe≈0.2	Pr=0.00	Pr=0.00	N/A	Weights: n x m Encoding: N/A Recall: feedforward

Note: (i) Memory configuration: Hopfield network: n=16. BAM, HK, and BP (no hidden layer): n=16, m=16, (ii) "Err–corre" is an abbreviation for error correction, (iii) both information capacity and error correction capability is measured under the minimum distance constraint (see section 2.3.2 and section 3.3.4) and the performance criterion – the probability of memory converging to stored patterns/associations is set to 0.95 and (iv) the performance of correcting error is measured under the condition: p=4 while the spurious and oscillatory states are measured under the condition: p=16 and Pe=0.

5. 2. Recommendations

Extending this work to other memory models and using a relatively larger size of network may constitute an important subject for further research. This work may focus on estimating the asymptotic information capacity for each tested model. Other research required to extend this work is to investigate how the quality of associative recall in the BP network is affected by the number of hidden neurons. The behavior of continuous value mapping within the interval [0, 1] for the BP network may be also worth investigating. Finally, the performance of adopting polynomial expansions of the input vector in higher order associative memory can be an interesting issue for future study.

5. 3. Reference

[MPRV1987]McEliece, R. J., Posner, E. C., Rodemich, E. R. and Venkatesh, S. S. (1987), "The capacity of the hopfield associative memory", IEEE Transactions on information Theory, Vol. 33, pp. 461–482.

APPENDIX I

AN EXAMPLE OF BIDIRECTIONAL RECALL IN BAM

The bidirectional search strategy used in BAM can not ensure improving the quality of associative recall. It is, sometimes, even detrimental to the performance, especially in the case of interpolative recall. One reason for this is that the Hebbian learning rule can not guarantee that all training pairs are at local energy minima. On the other hand, the fact that the bidirectional search process always seeks the state which has a lower energy potential also results in the training pair with relative higher energy potential seldom being recalled. The following example shows that BAM fails to recall the second training pair even if the memory is evoked by the training pair itself. Here, the BAM contains four input neurons and three output neurons. The number of stored patterns is three.

Table I BAM Training Patterns

No.	Input patterns	Oouput patterns	Energy
1	−1 −1 −1 −1	−1 1 −1	−14
2*	−1 −1 −1 1	1 −1 1	−10
3	1 −1 1 −1	1 −1 −1	−12

Note: BAM fails to recall the second training pair (marked by *).

Table II BAM Weight Matrix $\{ w_{ji} \}$

	i				
j	1	2	3	4	θ
1	–1	–3	1	–1	0
2	1	3	–1	1	0
3	–3	–1	–1	1	0

Note: weight matrix $\{ w_{ij} \}$ is the transposed form of $\{ w_{ji} \}$

Table III Intermediate States in Bidirectional Recall

Iterations	X space	Mapping	Y space	Energy
1	–1 –1 –1 1	→	1 –1 1	–10
	–1 –1 1 –1	←	1 –1 1	–14
2	–1 –1 1 –1	→	1 –1 1	–14 (stable)

Figure I depicts the results of the investigation taken on two different types of recall, i.e., the unidirectional and bidirectional recall. As it is shown, in most cases, the probability of bidirectional recall is lower than that of unidirectional recall.

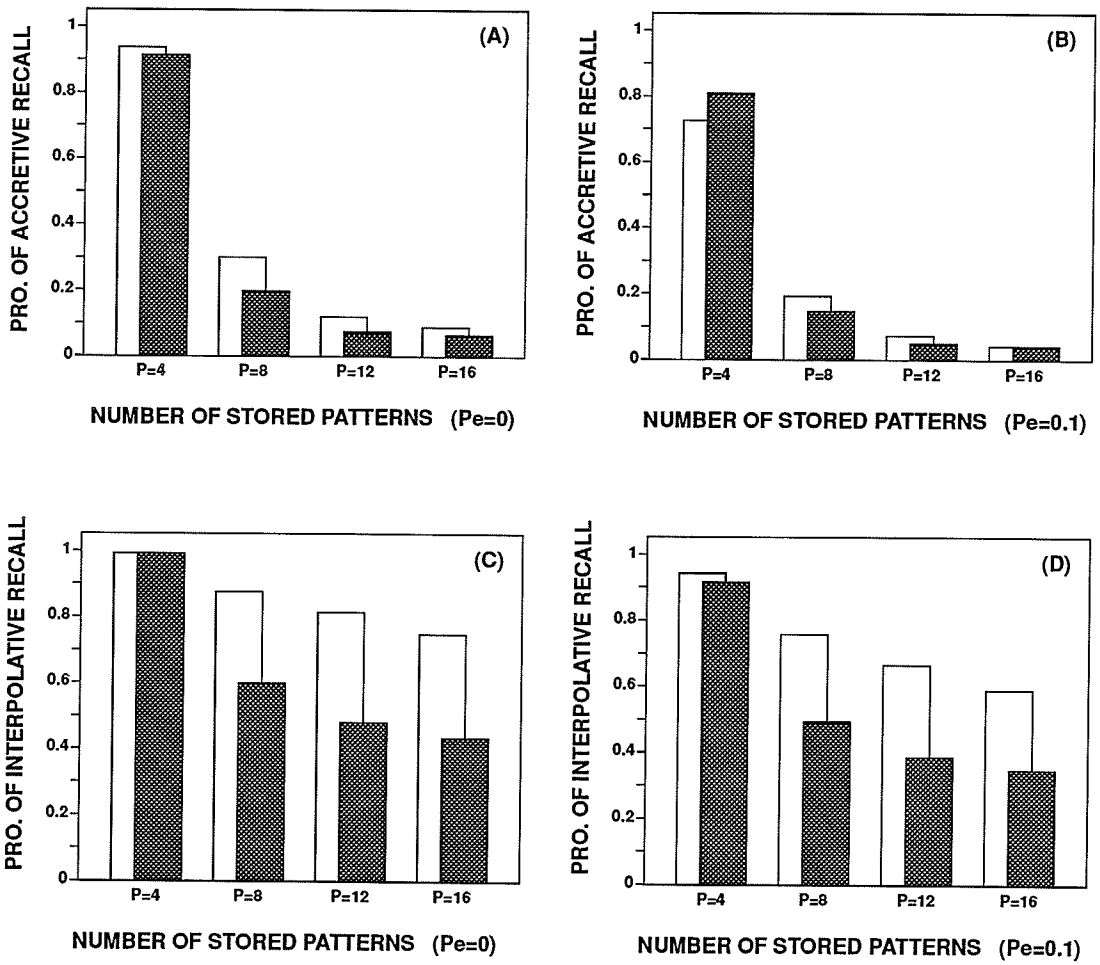

Fig. I Performance comparison between unidirectional recall and bidirectional recall. Tested models were configured as: n=16, m=16. ☐ UHAM (Unidirectional heteroassociative associative memory implemented by Hebbian rule), ▩ BAM. Fig. (a) and Fig. (b): accretive recall, Fig. (c) and Fig. (d): interpolative recall.

APPENDIX II

WEIGHTS – THEIR FLEXIBILITY

Weights in BAM, HK, and BP memories are presented in this appendix. Table (IV) is the set of (bipolar) training pairs attempted to be stored in BAM. BAM contains eight inputs neurons and eight output neurons. The number of patterns stored in BAM is four. The weights in BAM are listed in Table V. Similar training pairs were applied to encoding HK and BP networks, but all "–1" were replaced by "0" (see Table VI). The real–value weights for the HK and BP networks are listed in Table VII– IX. From this typical example, one can see that the BP algorithm provides the most flexible weights and thresholds. These weights take 69 different real values within the range [–3.79, 3.38]. The next is the Ho–Kashyap algorithm. It generates 30 and 22 different values for $\{ w_{ji} \}$ and $\{ w_{ij} \}$ within the range [–0.57, 0.46] and [–0.38, 0.43] respectively. The least flexible weights are generated by the BAM encoding algorithm. The integer weights take only 5 different values within the range [–4, 4].

Table IV BAM Training Patterns

No.	Input patterns	Output patterns
1	–1 –1 –1 1 1 1 –1 1	1 –1 –1 –1 –1 1 1 1
2	1 –1 1 1 1 1 1 –1	–1 1 1 –1 1 1 1 1
3	1 1 –1 –1 1 1 1 1	1 1 1 –1 –1 –1 1 –1
4	–1 1 1 1 –1 –1 1 –1	–1 1 1 1 –1 –1 –1 1

Table V BAM Weight Matrix $\{ w_{ji} \}$

	i								
j	1	2	3	4	5	6	7	8	θ
1	0	0	–4	–2	2	2	–2	4	0
2	2	2	2	0	0	0	4	–2	0
3	2	2	2	0	0	0	4	–2	0
4	–2	2	2	0	–4	–4	0	–2	0
5	2	–2	2	0	0	0	0	–2	0
6	0	–4	0	2	2	2	–2	0	0
7	2	–2	–2	0	4	4	0	2	0
8	–2	–2	2	4	0	0	0	–2	0

Note: weight matrix $\{ w_{ij} \}$ is the transposed form of $\{ w_{ji} \}$

Table VI HK and BP Training Patterns

No.	Input patterns								Output patterns							
1	0	0	0	1	1	1	0	1	1	0	0	0	0	1	1	1
2	1	0	1	1	1	1	1	0	0	1	1	0	1	1	1	1
3	1	1	0	0	1	1	1	1	1	1	1	0	0	0	1	0
4	0	1	1	1	0	0	1	0	0	1	1	1	0	0	0	1

Table VII HK Model Weight Matrix $\mathbf{x} \rightarrow \mathbf{y}$ $\{ w_{ji} \}$

	i								
j	1	2	3	4	5	6	7	8	θ
1	–0.11	0.11	–0.37	–0.16	0.11	0.11	–0.16	0.37	0.05
2	0.22	0.28	0.16	–0.04	0.03	0.03	0.46	–0.16	0.26
3	0.22	0.28	0.16	–0.04	0.03	0.03	0.46	–0.16	0.26
4	–0.18	0.18	0.11	–0.03	–0.32	–0.32	–0.03	–0.11	–0.16
5	0.45	–0.45	0.32	–0.08	0.05	0.05	–0.08	–0.32	–0.47
6	0.07	–0.57	0.11	0.22	0.18	0.18	–0.28	–0.11	–0.16
7	0.18	–0.18	–0.11	0.03	0.32	0.32	0.03	0.11	0.16
8	–0.28	–0.22	0.26	0.46	0.03	0.03	–0.04	–0.16	0.26

Table VIII HK Model Weight Matrix $\mathbf{y} \rightarrow \mathbf{x}$ $\{ w_{ij} \}$

	j								
i	1	2	3	4	5	6	7	8	θ
1	0.00	0.29	0.29	−0.29	0.43	−0.00	0.29	−0.43	−0.14
2	0.08	0.17	0.17	0.17	−0.25	−0.42	−0.17	−0.25	−0.00
3	−0.42	0.17	0.17	0.17	0.25	0.08	−0.17	0.25	0.00
4	−0.25	−0.07	−0.07	0.07	−0.11	0.25	−0.07	0.61	0.29
5	0.17	0.05	0.05	−0.38	0.07	0.17	0.38	−0.07	0.14
6	0.17	0.05	0.05	−0.38	0.07	0.17	0.38	−0.07	0.14
7	−0.17	0.38	0.38	−0.05	0.07	−0.17	0.05	−0.07	0.14
8	0.42	−0.17	−0.17	−0.17	−0.25	−0.08	0.17	−0.25	0.00

Table IX BP network Weight Matrix $\mathbf{x} \rightarrow \mathbf{y}$ $\{ w_{ji} \}$

	i								
j	1	2	3	4	5	6	7	8	θ
1	−0.57	0.20	−3.34	−1.15	1.25	0.65	−1.19	3.38	0.28
2	2.34	1.89	1.72	−0.90	−0.77	−1.03	3.23	−1.24	−0.21
3	2.11	1.91	2.08	−1.25	−0.52	−0.61	3.27	−1.70	−0.37
4	−1.37	1.59	1.49	0.35	−2.86	−3.25	0.36	−1.26	0.91
5	1.63	−3.79	1.18	−0.33	0.55	0.52	0.25	−3.41	−0.86
6	0.20	−4.73	0.53	1.82	1.69	0.92	−1.36	−0.44	−0.04
7	1.66	−1.61	−1.46	−0.72	3.38	2.41	−0.73	1.14	−0.43
8	−1.37	−1.62	2.62	4.66	0.15	−0.65	−0.73	−0.49	0.45

APPENDIX III

PERFORMANCE OF THE HOPFIELD MODEL RECALLING BOTH ORIGINAL AND REVERSED TRAINING PATTERNS

The error correction performance of the Hopfield network tested under the condition that the reversed version of training patterns are allowed for representing the original training patterns is presented. This performance is then compared with the results presented in section 3.3.4 in chapter 3. As illustrated in Fig. II, the probability of interpolative recall

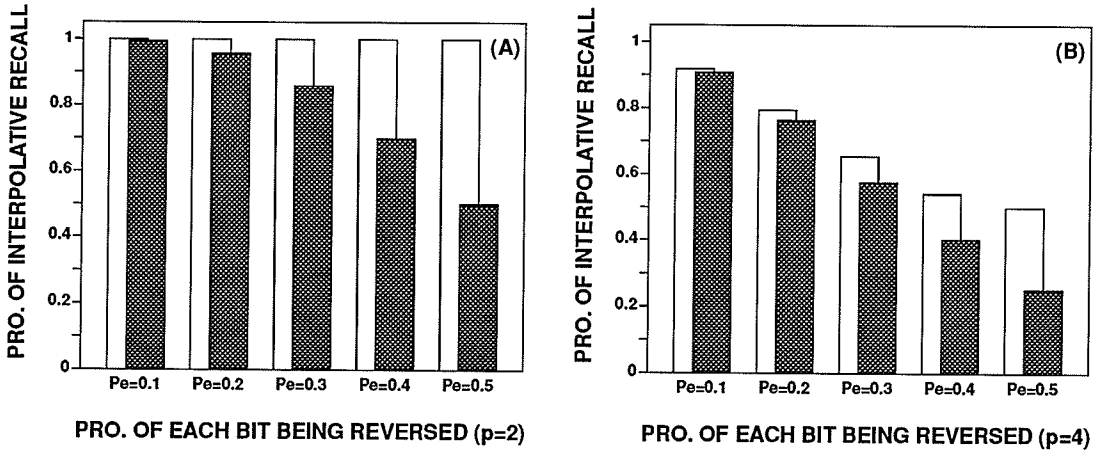

Fig. II Performance of Hopfield network recalling both original and reversed training patterns. ☐ *Model I*: the reversed version of training patterns are allowed for representing the original training patterns. ▩ *Model II*: only training patterns are permitted to represent the correct information stored in the memory. Network structure: n=16. Testing patterns: bipolar mode.

tested under this newly defined condition is higher compared with the results presented in section 3.3.4. These differences become more pronounced as the noise level increases. It

should be pointed out that the maximum pattern with which the n neurons can represent is narrowed to $\frac{1}{2}2^n$ if the new definition is adopted. In this experiment, the randomly generated training patterns strictly follow this definition, i.e., the pattern was not allowed to participate in the training pair if its reversed version had already been generated.