PATTERN CLASSIFICATION AND ASSOCIATIVE RECALL BY NEURAL NETWORKS

Thesis by

Tzi-Dar Chiueh

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

1989

(Submitted May 24, 1989)

To My Beloved Parents and My Dearest Wife

Acknowledgements

The author would like to express his gratitude to Professor Rodney M. Goodman for his guidance and interest in this work. Many helpful and stimulating discussions with Amir Atiya and John Miller are also acknowledged. Thanks are also due to Patrick Smythe, Sean Coffey, Kathleen Kramer, and Chi-Chao Chao. The author acknowledges financial support of two teaching assistantships and various research assistantships from the Institute.

Last, but not least, the author wishes to gratefully acknowledge his parents Mr. C. M. Chiueh and Ms. Y. J. W. Chiueh, and his wife Jill J. H. Wang for their encouragement and support during his period of residence at the Institute.

Abstract

The first part of this dissertation discusses a new classifier based on a multilayer feed-forward network architecture. The main idea is to map irregularly-distributed prototypes in a classification problem to codewords that are organized in some way. Then the pattern classification problem is transformed into a threshold decoding problem, which is easily solved using simple hard-limiter neurons. At first we propose the new model and introduce two families of good "internal representation" codes. Then some analyses and software simulation concerning the storage capacity of this new model are done. The results show that the new classifier is much better than the classifier based on the Hopfield model in terms of both the storage capacity and the ability to classify correlated prototypes.

A general model for neural network associative memories with a feedback structure is proposed. Many existing neural network associative memories can be expressed as special cases of this general model. Among these models, there is a class of associative memories, called correlation associative memories, that are capable of storing a large number of memory patterns. If the function used in the evolution equation is monotonically nondecreasing, then a correlation associative memory can be proved to be asymptotically stable in both the synchronous and asynchronous updating modes.

Of these correlation associative memories, one stands out because of its VLSI implementation feasibility and large storage capacity. This memory uses the exponentiation function in its evolution equation; hence it is called exponential correlation associative memory (ECAM). It is further proved that the storage capacity of ECAM scales exponentially with N (the number of components in memory patterns) when N approaches infinity. A programmable ECAM chip is designed, simulated, fabricated, and then tested. The performance of the ECAM chip is shown to be not much worse than that of a computer-simulated ECAM model in terms of error correcting ability (attraction radius). Finally, the speed of the prototype ECAM chip is demonstrated by employing it to do vector quantization on binary images. And it is found that the ECAM chip can process binary images in real time.

Contents

List of Figures

List of Tables

Chapter 0

Overview

The resurgence of neural network research in these past few years has been phenomenal. Many researchers have done much investigation on neural networks with a view to using them for solving difficult problems, such as diagnosis, prediction, motor control, pattern classification, combinatorial optimization, associative recall, etc. In this dissertation, by "neural networks" we mean massively parallel, analog computation systems consisting of simple processors (neurons) together with certain construction methods for system parameters, such as connection weights among neurons and thresholds of neurons. We are interested in analog computation systems primarily because we feel that most potential applications of neural networks are those problems that require fast, yet fuzzy computation, which is just what analog computation systems can provide.

Of the many promising applications of neural networks, we are particularly interested in pattern classification and associative recall. The simplest form of the pattern classification problem is the minimum distance classification problem, i. e., the problem when conditional probability distributions of all classes are assumed to be multivariate Gaussian and all features uncorrelated. This problem has a very close tie with the associative recall problem because in both cases one needs to find the nearest prototype (memory pattern) to the input according to some distance measure. These two problems have attracted much attention for a few decades, and there has already been much success in solving them by digital computers. Still, we want to explore neural network approaches of solving these two problems because we believe that with the speed and hardware efficiency of massively parallel, analog computation systems, pattern classification and associative recall can be done more effectively.

Hopfield proposed an innovative neural network model as an associative memory; however, it has been reported that his model has some major handicaps — limited storage capacity and excessively large hardware complexity. Since then, many attempts have been made to remedy these drawbacks. Nevertheless, many of these new models e. g., potential function associative memory, were designed without consideration of implementation issues, so that building such models becomes a formidable challenge. Therefore, most of them were only simulated by computer program and were never realized in hardware (optics or silicon). But as just mentioned, the expedition of analog computation is one of the most important reasons that make neural network approaches appealing; we thus feel that any good neural network model must be suitable for efficient hardware realization.

The first part of this dissertation discusses a new classifier based on a multilayer feed-forward network architecture. The main idea is to map irregularly-distributed prototypes in a classification problem to codewords that are organized in some way. Then the pattern classification problem is transformed into a threshold decoding problem, which is easily solved using simple hard-limiter neurons. At first we propose the new model and introduce two families of good "internal representation" codes. Then some analyses and software simulation concerning the storage capacity of this new model are done. The results show that the new classifier is much better than the classifier based on the Hopfield model in terms of both the storage capacity and the ability to classify correlated prototypes.

A general model for neural network associative memories with a feedback structure is proposed. Many existing neural network associative memories can all be expressed as special cases of this general model. With this model, new associative memories suitable for implementation in other technologies can easily be introduced. A class of associative memories, called correlation associative memories (CAM), can also be expressed as instances of this general model. It is also proved that if the function used in the evolution equation of a particular CAM is monotonically nondecreasing, then that CAM is asymptotically stable in both synchronous and asynchronous updating modes.

Of these correlation associative memories, one stands out because of its VLSI implementation feasibility and large storage capacity. This memory uses the exponentiation function in its evolution equation; hence, it is called exponential correlation associative memory (ECAM). Furthermore, it is proved that ECAM has a storage capacity that scales exponentially with N (the number of components in memory patterns) when N approaches infinity. A programmable ECAM chip is designed, simulated, fabricated, and then tested. We find that the performance of the ECAM chip is almost as good as that of a computer-simulated ECAM system in terms of attraction radius. Finally the speed of the chip is measured by applying it to executing vector quantization on binary images. And it is found that the ECAM chip is so fast that it can process binary images in real time.

Chapter one serves as a summary of a comprehensive literature survey about traditional (digital sequential computer) and neural network approaches of pattern classification and associative recall. As mentioned earlier, we are interested in neural network approaches because we want to take advantage of the speed and hardware efficiency of analog computation systems. The first half of Chapter one reviews the general status of computer algorithms for pattern classification; corresponding neural network approaches are also discussed whenever possible. The second half of Chapter one reports a survey of existing neural network associative memory models. In the last section of Chapter one, we discuss the relationship between a minimum distance pattern classifier and an associative memory, and we show how one can be built from the other. Furthermore, we give an account of drawbacks of existing winner-take-all circuits, which many researchers considered a good solution to pattern classification problems and associative recall problems.

In Chapter two, we suggest a new neural network classifier based on a feed-forward network with one hidden layer. The idea is to specify a set of "good" internal representations (activations of the hidden neurons) instead of using backpropagation to generate them. The motivation of this new classifier comes from the observation that, except for how the prototypes (codewords) are generated, minimum distance classification problems are similar to decoding problems in the error correcting code paradigm. In error correcting code problems, codewords are designed with a view to fast and simple decoding, while the prototypes in pattern classification problems are not designed and are distributed in the feature space randomly. Consequently, if one can map the prototypes in the feature space to codewords (internal representations) in the code space, the minimum distance classification problem becomes a simple decoding problem.

The new classifier uses the connection matrix from the input layer to the hidden layer to perform the mapping from the feature space to the code space. The connection weight matrix from the hidden layer to the output layer is responsible for threshold decoding. We propose two families of good codes for this new classifier — the maximal-length sequence codes and the Hadamard matrix codes. The storage capacity of the new classifier is shown to be larger than $0.22N$, where N is the number of bits in stored prototypes. We also run some simulation and find that the new classifier is much better than a classifier based on the Hopfield model when it comes to storage capacity and the ability to classify correlated prototypes.

In Chapter three, we introduce a general model for neural network associative memories with feedback structure. This general model subsumes most neural network associative memories with feedback architecture reported in Chapter one. It is based on an algorithm similar to the election process in political systems. At any timestep, the associative memory has a state, which is an N-bit binary pattern. To find the state of the next timestep, weighting functions f_k's defining the strengths of memory patterns are computed. Then a weighted sum of all memory patterns is calculated, and a decision is made to determine the polarity of each component in the next state pattern. This is not unlike a political election process, in which each city (component) has many political groups (memory patterns), each group has a different number of voters (strength or value of the weighting function) and party (polarity), and the next assembly (state pattern) is determined by holding a local election in all cities. We choose four models — the Kanerva memory, the BMW memory, the Hamming network associative memory, and the spectral associative memory as examples. And we show that, with proper choice of weighting functions, they can all be expressed as special cases of the general model. Moreover, the Kanerva memory and the BMW model are proved to be asymptotically stable provided that some strong assumptions are made.

Chapter four deals with correlation associative memories (CAMs) in general and the exponential correlation associative memory (ECAM) in particular. Correlation associative memories are models whose weighting functions depend only on some correlation measures of stored memory patterns and the state pattern of the system. We show that CAMs are also special cases of the general model in Chapter three. Furthermore, we prove that they are asymptotically stable in both synchronous and asynchronous updating modes if their weighting functions are monotonically nondecreasing.

In the second half of Chapter four, we concentrate on the exponential correlation associative memory, whose weighting functions f_k's are of the exponential form. We feel that ECAM is most amenable to VLSI implementation, since MOS transistors exhibit an exponential characteristic between the drain current and the gate-to-source voltage in the subthreshold range. We proved that for an ECAM with N-bit memory patterns, more than c^N memory patterns can be stored as N approaches infinity. The constant c is a parameter that depends on the recall error probability, the percentage error in the input patterns, and the base constant of the exponentiation function. However, it is to be noted that to build an ECAM storing exponential number of memory patterns, one also needs exponential hardware complexity.

A more important result is that for sufficiently large input percentage error, ECAM can store as many memory patterns as is allowed by the sphere-packing bound in information theory ($2^{N(1-\mathcal{H}(\rho))}$, where ρ is the input percentage error). We also consider the case when the exponentiation circuits of ECAM have limited dynamic range. It turns out that the storage capacity will then be proportional to the dynamic range. This finding is not necessarily discouraging in that the same conclusion about the high-order CAM has been reached. Finally, we present some computer simulation results showing that the previous theoretical predictions about the storage capacity of ECAM are valid even when N is less than 100.

In Chapter five, we are concerned with VLSI implementation of ECAM. The chip we design is a programmable associative memory chip based on the ECAM model. In the first part of the chapter, the design of a static RAM, which holds the information of stored memory patterns, is given. Next, analog computation circuits performing associative recall function are described. These circuits include 1) the correlation computation circuit; 2) the exponentiation, multiplication, and summing circuit; and 3) the thresholding circuit.

Next, we present the layout and the testing results of the ECAM chip. The final design of the ECAM chip is capable of storing 32 patterns, each 24 bits wide. After comprehensive testing, it is found that the ECAM chip fares almost as well as a computer-simulated ECAM with the base in the exponentiation function equal to two. As an application, we use the ECAM chip to solve the vector quantization problem of binary images, and we find that reconstructed images have fairly good quality. What is more important is that the ECAM chip can process images at a very high rate, more than 20 images per second.

Chapter 1

Pattern Classifiers and Associative Memories

1.1 Introduction

This chapter serves as the digest of a comprehensive literature survey about traditional (digital sequential computer) and neural network approaches of pattern classification and associative recall. We are interested in neural network approaches because of the speed and hardware efficiency provided by massively parallel, analog computation neural systems. The first half of this chapter describes the general status of how pattern classification is done by digital computer algorithms. Also, corresponding neural network approaches are introduced whenever possible. The second half of this chapter reports a survey of existing neural network associative memory models. Some traditional methods of doing associative recall, e. g., hash coding and its variations, are not discussed because they are not suitable for neural network implementation. In the last section, we discuss the relationship between a pattern classifier and an associative memory and show how one can be built from the other. Also, we give an account of the drawbacks of existing winner-take-all circuits, which many researchers claimed to be a viable solution to pattern classification problems and associative recall problems.

1.2 An Introduction to Pattern Classifiers

Pattern recognition is one of the most important subjects in information processing systems. It has application in various different areas, such as radar signal recognition, speech recognition, voice identification, production-line inspection, handwritten digit (character) recognition, medical diagnosis, sonar signal detection, vector quantization, and so on [52]. Because of its widespread usage, pattern recognition has received a great deal of attention during the past few decades. However, the problem itself, in its most general form, is "ill-defined" and thus is very hard. By "ill-defined problems," we mean problems whose answers are not definitive, such as pattern recognition problems or "Who is the best player in the National Basketball Association ?" Consider the following handwritten digit recognition problem. A symbol that looks like a "4" as much as a "6" is presented to a digit recognition system, then that system is asked to make a decision. How then will the performance of this system be determined ? Obviously, the answer lies in the opinions of those involved in judging this system. Since each different individual has a different perception of what a "4" or a "6" should look like, it will be very difficult, if not impossible, to reach a consensus. Hence, a definitive performance measure of pattern recognition systems is hard to come by, and we say that the pattern recognition problems are ill-defined.

We will, in this dissertation, regard patterns as vectors of real numbers, which are measurements of physical quantities in some environment. Examples of patterns are : arrays of pixels (picture elements), sampled speech waveforms, symptoms of patients, etc. Figure 1.1 illustrates a typical pattern recognition system, which consists of four major components [14, 16]. The data-acquisition block is responsible for capturing signals from the outside environment and converting them to a form that can be further processed by the following components in the pattern recognition system. For instance, a camera is an appropriate device for acquiring images in a production-line inspection system, while a microphone is the proper choice for converting speech to electrical signals in a voice identification system. Generally speaking, the amount of information in the

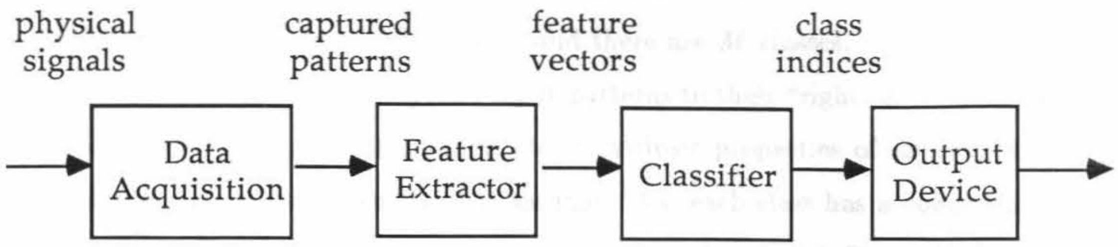


Figure 1.1: Block diagram of a typical pattern recognition system

patterns is usually too enormous to allow real-time processing. In addition, there is much redundancy in the patterns. As a result, most pattern recognition systems adopt an approach called *feature extraction* to condense the information conveyed in an input pattern to an N-dimensional feature vector. After this stage, the classifier then works on the feature vector and determines which category the input pattern belongs to. The index of that category is then passed on to the output device, which manipulates the decision and displays it in some format.

The design of any pattern recognition system must take into account the construction of its feature extractor and classifier. Very often, the efficiency of one has much influence on the other. For example, if a feature extractor can generate a feature that is exactly the class index of an output pattern, then no classification is necessary and the recognition is done. On the other hand, if a classifier is so powerful that it can process an enormous amount of information and make the right decision relying solely on input patterns, then that pattern recognition system can do without a feature extractor. As a matter of fact, the design process of a pattern recognition system usually involves an iterative procedure, which goes back and forth improving the feature extractor and the classifier until a satisfactory performance is attained. Nevertheless, feature extraction depends heavily on the specific problem at hand; therefore, we will concentrate on classifier design only.

Let us now formulate rigorously the problem of pattern classification. In a pattern classification problem, inputs are N-dimensional feature vectors (for brevity, we call them patterns), each feature is a real number, and there are M classes, $C^{(1)}, C^{(2)}, \cdots, C^{(M)}$. The goal of a classifier is to categorize input patterns to their "right" classes. To achieve this goal, a classifier need information about intrinsic properties of all classes and interactions among classes. Usually, it is assumed that each class has a conditional probability distribution over the sample space $\mathbf{R}^N$; i. e., there exist $P_k(\mathbf{z}) : \mathbf{R}^N \rightarrow \mathbf{R}$, for $k = 1, 2, \ldots, M$ and

$$\int_{\mathbf{R}^N} P_k(\mathbf{z})\, d\mathbf{z} = 1, \qquad\qquad k = 1, 2, \ldots, M,$$

where $\mathbf{R}$ is the field of real numbers. Furthermore, since results for the equal *a priori* probabilities case are easily generalized to the case when not all M *a priori* probabilities are equal, input patterns are assumed to occur from these M classes with equal *a priori* probabilities. In order to minimize the probability of making an incorrect classification, the classifier first calculates $P_k(\mathbf{x})$, for $k = 1, 2, \ldots, M$, where $\mathbf{x}$ is the input pattern, then finds the maximum of these M conditional probabilities, and categorizes the input pattern to the class with maximum conditional probability. The above procedure for finding the right class for an input pattern is called the *Bayes decision rule.* Unfortunately, the conditional probability distributions $P_1(\mathbf{x}), P_2(\mathbf{x}), \cdots, P_M(\mathbf{x})$ are usually not known; therefore, the primary duty of a classifier designer is to estimate these conditional probability distributions with the help of a set of sample input patterns drawn from these M classes (*training set*).

1.3 Design and Implementation of Pattern Classifiers

There has been extensive exploration of many pattern classifiers using the traditional (digital computer) methods [7, 14, 15, 16, 20, 49, 59, 64]. In this section, we will summarize these traditional approaches for solving pattern classification problems and give the corresponding neural network implementation whenever possible.

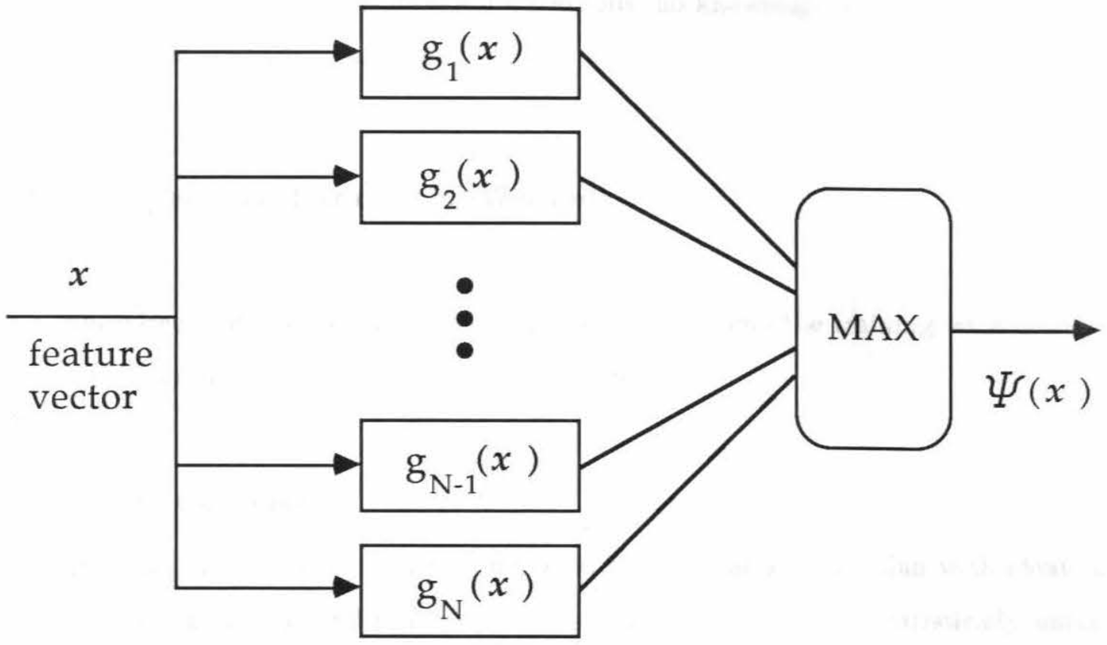


Figure 1.2: Architecture of classifiers based on the discriminant function method

Before we present different types of pattern classifiers, let us introduce the idea of *discriminant function* [14, 16]. Of the many ways to represent classifier designs, the most common one is through a set of discriminant functions, $g_k(\mathbf{x}),\ k = 1, 2, \ldots, M$, one for each class. In classification systems based on discriminant functions (see Figure 1.2), the classifier decides that the input pattern $\mathbf{x}$ belongs to class j if

$$g_j(\mathbf{x}) > g_k(\mathbf{x}) \qquad \forall\ k = 1, 2, \ldots, M, \text{and } k \neq j. \tag{1.1}$$

The design method of classifiers can be either *supervised* or *unsupervised*. The former is the case when all sample patterns in the training set are labeled with the indices of the classes they are drawn from, while in the latter case no labeling is available. Another dichotomy of classifier design methods is *parametric* versus *nonparametric*. In parametric methods, it is assumed that the form of the conditional probability distributions is known, and the designer has to estimate parameters of those distributions from the training set. While in nonparametric methods, no knowledge about the conditional probability distributions is assumed.

1.3.1 Supervised Parametric Methods

In a supervised, parametric classifier design method, not only the training set is labeled, but also the form of $P_k(\mathbf{x})$ is assumed to be known.

- **Linear Classifiers**

  If $P_k(\mathbf{x})$, $k = 1, 2, \ldots, M$ are assumed to be multivariate Gaussian with identical covariance matrix, and if all N features in input patterns are statistically uncorrelated, then finding the "right" class of an input pattern $\mathbf{x}$ becomes finding the maximum of

$$g_k(\mathbf{x}) \equiv -(\mu_k - \mathbf{x})^t (\mu_k - \mathbf{x}), \qquad k = 1, 2, \ldots, M, \tag{1.2}$$

  where μ_k is the mean feature vector ("prototype") of samples labeled $C^{(k)}$ in the training set.

  Since the discriminant functions in Equation (1.2) can usually be expressed in linear forms, these classifiers are usually called *linear classifiers, correlation classifiers,* or *template-matching classifiers* [14]. Furthermore, since the discriminant function is proportional to the square of the Euclidean distance between μ_k and $\mathbf{x}$, these classifiers are also called *minimum Euclidean distance classifiers.* If the input features are binary instead of real; i. e., when the input pattern space is $\mathbf{B}^N \equiv \{-1, 1\}^N$, then these classifiers become *minimum Hamming distance classifiers,* and $g_k(\mathbf{x}) = -4\, d_{\text{Hamming}}(\mu_k, \mathbf{x})$. Linear classifiers can be realized by a one-layer neural network followed by a winner-take-all circuit as suggested by Lippmann [39].

- **Quadratic Classifiers**

If all M conditional probability distributions are multivariate Gaussian but with different covariance matrices, then the discriminant functions become quadratic and

$$g_k(\mathbf{x}) \;\equiv\; -(\mu_k - \mathbf{x})^t\, \mathbf{\Sigma}_k^{-1}\, (\mu - \mathbf{x}_k), \qquad\qquad k = 1, 2, \ldots, M, \qquad (1.3)$$

where $\mathbf{\Sigma}_k$ is the covariance matrix of the set of samples labeled $C^{(k)}$. This type of classifiers can be implemented by a one-layer, high-order neural network [17, 38] followed by a winner-take-all circuit [39].

- **Piecewise Linear Classifiers**

When some $P_k(\mathbf{x})$'s are multimodal, it is desirable to use piecewise linear functions as class boundaries. A discriminant function is said to be piecewise linear if its output is obtained by a "max" operation [14]. Suppose class $C^{(k)}$ has n_k subclasses, each with prototype μ_{kj}, $j = 1, 2, \ldots, n_k$; then the discriminant functions become piecewise linear,

$$g_k(\mathbf{x}) \;\equiv\; \max_j \left\{ -(\mathbf{x} - \mu_{kj})^t\, \mathbf{\Sigma}_{kj}^{-1}\, (\mathbf{x} - \mu_{kj}) \right\}. \qquad\qquad k = 1, 2, \ldots, M, \quad (1.4)$$

This type of classifiers can be implemented by a one-layer, high-order neural network [17, 38] followed by two layers of winner-take-all circuits.

- **Classifiers with Internal Representation Codes**

Since there are many drawbacks in winner-take-all circuits (for detail, see Section 1.6), there has been much research on building classifiers without winner-take-all circuits. We proposed a two-layer neural network classifier utilizing an internal coding scheme that transforms minimum distance classification problems to threshold decoding problems ([9]; also see Chapter two). Various other models taking advantage of coded internal representations have also been suggested, and all of them showed satisfactory results [6, 27, 31].

1.3.2 Supervised Nonparametric Methods

In this type of classifier design methods, no assumption is made on the form of the M conditional probabilities; so different approaches are taken to design classifiers. In the following discussion, only the two-category case is discussed since the multicategory case can be obtained by generalization.

- **Perceptron with Perceptron Learning Rule**

  The architecture of this type of classifiers is a one-layer perceptron with one hard-limiter neuron, whose output is "-1" if the activation level at the input is negative, "$+1$" otherwise. The discriminant function used is

$$g(\mathbf{x}) \equiv \mathbf{w}^t \mathbf{x}. \tag{1.5}$$

  The coefficients in the $N \times 1$ vector $\mathbf{w}$ are to be estimated from the training set in such a way that for each $\mathbf{x}_j$ labeled class $C^{(1)}$, $\mathbf{w}^t \mathbf{x}_j > 0$, and for each $\mathbf{x}_j$ labeled class $C^{(2)}$, $\mathbf{w}^t \mathbf{x}_j < 0$, or equivalently, $-\mathbf{w}^t \mathbf{x}_j > 0$. Accordingly, if all samples labeled $C^{(2)}$ in the training set are replaced by their negative counterparts, correct classification on all samples in the training set is obtained if

$$\mathbf{w}^t \mathbf{x} > 0 \qquad\qquad \text{for all } \mathbf{x} \text{ in the new training set.} \tag{1.6}$$

  Rosenblatt proposed a way of successively updating the coefficients of $\mathbf{w}$ with a view to satisfying the above inequality [55]. It was also shown that this updating procedure will eventually converge to a vector $\mathbf{w}_o$ that satisfies the inequality in Equation (1.6) as long as such a vector exists, namely, as long as sample patterns of these two categories in the training set can be separated by a straight line ("Perceptron Convergence Theorem" [14, 47]).

- **Perceptron with Other Minimum Squared Error Procedures**

Instead of the inequality in Equation (1.6), one can require that the following equation be satisfied

$$\mathbf{w}^t \mathbf{X} = \mathbf{y}^t, \tag{1.7}$$

where columns in $\mathbf{X}$ are samples in the training set and all components of $\mathbf{y}$ are positive. Since the dimension of $\mathbf{y}$ is usually very large when compared to the dimension of $\mathbf{w}$, it is usually difficult to find a $\mathbf{w}$ that satisfies Equation (1.7) exactly. Therefore, a few methods minimizing the sum of squared errors have been suggested, such as the pseudoinverse method [32, 33], Widow-Hoff procedure [67], and Ho-Kashyap method [23]. These classifiers can all be realized by a one-layer perceptron.

- **Multilayer Feed-forward Neural Networks**

Lippmann showed that a three-layer perceptron consisting of hard-limiter neurons can produce class regions of any shape [39, 40, 27]. For the sake of applying gradient descent, Rumelhart and his colleagues adopted neural networks made up of neurons with sigmoidal response instead. They proposed a training algorithm that is a gradient descent method minimizing a cost function — the sum of squared output errors of all input-output pairs in the training set [56]. This procedure, called *error backpropagation* or *generalized delta rule*, has been shown to provide good performance in solving many different pattern recognition problems [1, 28, 35, 37, 40, 50, 60, 66, 68]. Nevertheless, there is one big disadvantage, which often renders the backpropagation method awkward, if not useless : The amount of time it takes to train a particular neural network is very long, and what's worse, as the training set grows larger, the training time seems to grow faster than linear with the training set size.

- **Parzen Window Method and K-Nearest-Neighbor Method**

  In the Parzen window approach, the continuous probability distribution $P_k(\mathbf{x})$ is constructed by summing N-dimensional *window functions* centered at those positions where sample patterns from $C^{(k)}$ lie. These window functions approach the N-dimensional *Dirac delta function* as the size of the window functions shrinks. After all $P_k(\mathbf{x})$'s are constructed, Bayes decision rule can be applied to do classification. The problem with the Parzen window approach is that it requires a very large training set in order to get satisfactory recognition performance.

  As for the K-nearest-neighbor method, the formula is : Given an input pattern $\mathbf{x}$, find the K nearest neighbors of $\mathbf{x}$ in the training set, then count the number of nearest neighbors for each class, and categorize $\mathbf{x}$ to the class that has the largest number of nearest neighbors. The main disadvantage of this method is that the whole training set needs be stored during classification, which is very undesirable. The Reduced (Restricted) Coulomb Energy (RCE) network proposed by a group led by Cooper [54, 58] is the neural network implementation of a variation of the K-nearest-neighbor method.

1.3.3 Unsupervised Methods

The only difference between supervised learning and unsupervised learning is that in unsupervised learning methods, patterns in the training set are not labeled. A simple strategy for solving unsupervised classifier design problems is to apply clustering procedures to the training set and then form clusters of patterns so that all patterns in a cluster are close to one another in some similarity measure. All patterns in the training set can be labeled; then a supervised learning method can be applied to the newly labeled training set. In this subsection, we will concentrate only on clustering techniques.

- **K-Mean Clustering Algorithm and ISODATA**

The K-mean clustering algorithm works as follows : At first, choose K initial cluster means randomly from the training set. Next, for each sample in the set, find the nearest cluster mean in some similarity measure, label that sample with the index of that cluster, and then compute the new cluster mean. The algorithm terminates when all K means stay unchanged after one run through the training set.

The ISODATA (Iterative Self-Organization Data Analysis Technique A) is similar to the K-mean method, except that ISODATA employs more "heuristics," which are results of experience gained through experimentation. With those heuristics, ISODATA can merge, discard, or split clusters whenever necessary. Another variation of the K-mean method is the Kohonen's self-organizing feature map technique [33], in which case means of neighboring clusters are also updated.

- **Leader Clustering Algorithm**

The leader clustering algorithm introduced here is a somewhat sophisticated version of the original leader clustering method [22]. In the leader clustering algorithm, an input pattern in the training set is presented to the classifier, the similarity measures of this pattern with all cluster means are computed, and the cluster with maximum similarity measure is activated, while all the other clusters are suppressed. If the maximum similarity measure is not large enough, create a new cluster with the current input pattern as its cluster mean. Otherwise the input pattern has to pass a "vigilance test," which checks how close the input pattern is to the cluster mean of the activated cluster. If the input pattern fails the vigilance test, it is checked for vigilance against the cluster with the next largest similarity measure, etc. If the input pattern passes the vigilance test, the cluster mean of that particular cluster is recomputed, taking the input pattern into account. The algorithm then goes back to process the next pattern in the training set. This algorithm is realized by the Adaptive Resonance Theory (ART) model introduced by Carpenter and Grossberg [8, 21, 45]. The major drawback of ART model is that it can perform well with perfect input patterns, but even a small amount of noise can cause problems [39].

- **Other Clustering Techniques**

  There are various other interesting clustering techniques, such as the cluster seeking algorithm, the maximin distance clustering method, hierarchical clustering algorithms, graph theoretic clustering algorithms, and so forth [7, 14, 29, 64]. Since we cannot find any neural implementation for them, we will not discuss them here.

1.4 An Introduction to Associative Memories

Memory plays a very crucial role in information processing of both biological organisms and digital computers. In a system of the Von Neumann type, an "addressing" memory (listing memory) works with a central processing unit in order to execute useful operations. By addressing memories, we mean memories whose data items are accessed by providing their addresses. The Von Neumann model, albeit effective, has been known to suffer from a bottleneck between the central processing unit and the system memory. One of the reasons is that as the VLSI fabrication technology advances, the speed of CPUs has grown phenomenally, while commercial memory chips still cannot run as fast as CPUs. For instance, the latest Intel 80860 can run at a speed of 40 MHz or 25 ns cycle time, while commercial DRAM chips have a typical access time of about 80 ns. Another factor is that as computers grow more powerful, more information goes through the channel between the CPU and the system memory, so the channel becomes a bottleneck. The human brain, on the other hand, distributes processing among many simple processors (neurons), which process information *in situ*. Collectively, these processors perform complicated functions, such as reasoning, mobile control, recognition, etc.

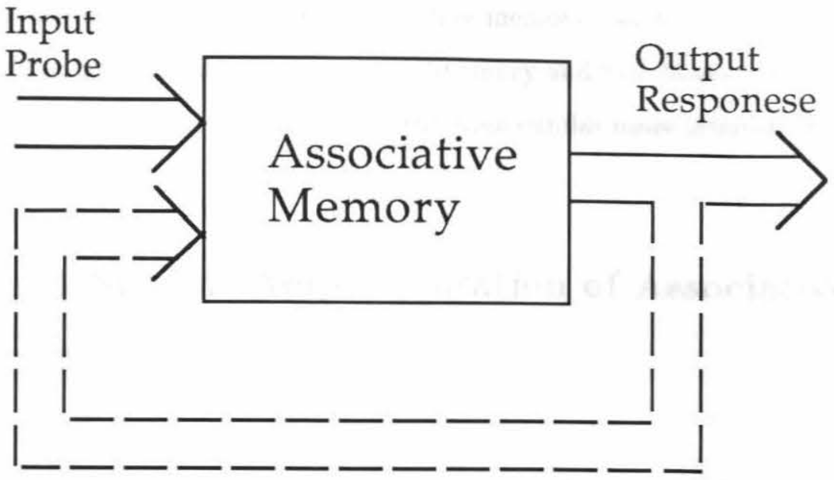


Figure 1.3: A model of associative memories

In Figure 1.3, we delineate a simple model for associative memories. Suppose a set of M key-response pairs $\{(\mathbf{u}^{(k)}, \mathbf{z}^{(k)})\}$ has been stored in the associative memory. The function of the associative memory is : Given an input pattern $\mathbf{x}$, find the nearest key $\mathbf{u}^{(j)}$ in some metric (similarity measure) and put out the corresponding response $\mathbf{z}^{(j)}$. There are, however, many possible metrics. The one most often used is Euclidean distance if input patterns have real components and Hamming distance if input patterns have binary components.

There are basically two different types of associative memories : *autoassociative memory* and *heteroassociative memory*. The former refers to those associative memories with identical key patterns and response patterns, i. e., $\mathbf{u}^{(k)} \equiv \mathbf{z}^{(k)},\ k = 1, 2, \ldots, M$, and the latter otherwise. The dashed feedback path is desirable in autoassociative memory system. With that path, the system need not generate a perfect recall at once. Instead, it can first produce a partially correct response, feed it back to the input end as the key to the system again, and improve the response gradually through iteration to obtain a complete recall.

In this dissertation, only autoassociative memories will be explored. The reason for this decision is twofold : 1) Any heteroassociative memory can be built by cascading a look-up table to an appropriate autoassociative memory and 2) autoassociative memories are mostly systems with feedback loops and thus often exhibit more interesting behaviors.

1.5 Neural Network Implementation of Associative Memories

Since associative recall is a vital aspect of human brain functions, it is no surprise that neural network research has played a major role in the study of associative memory. Earlier neural network associative memory models can be found in various references [3, 32]. Although many potential applications have been suggested and researched, these models have neither drawn much attention nor made any major breakthrough. By the time when digital computer technology started to take off, most attention was focused on the "addressing" memory systems compatible with digital computers, e. g., RAMs, ROMs.

It was not until 1982, when Hopfield [24] debuted his associative memory model, which is based on the sum-of-outer-product construction rule and the energy-minimizing principle, that neural network associative memories began to attract attention again. Since then, many researchers have tried, with a certain degree of success, to improve the Hopfield model with a view to overcoming the limited storage capacity problem. In the remainder of this section, a review of important neural network associative memory models will be presented. Before we proceed, let us define some terms : Let N be the number of components in the patterns, M be the number of memory patterns stored in the autoassociative memory, and $\mathbf{u}^{(1)}, \mathbf{u}^{(2)}, \cdots, \mathbf{u}^{(M)}$ be those memory patterns.

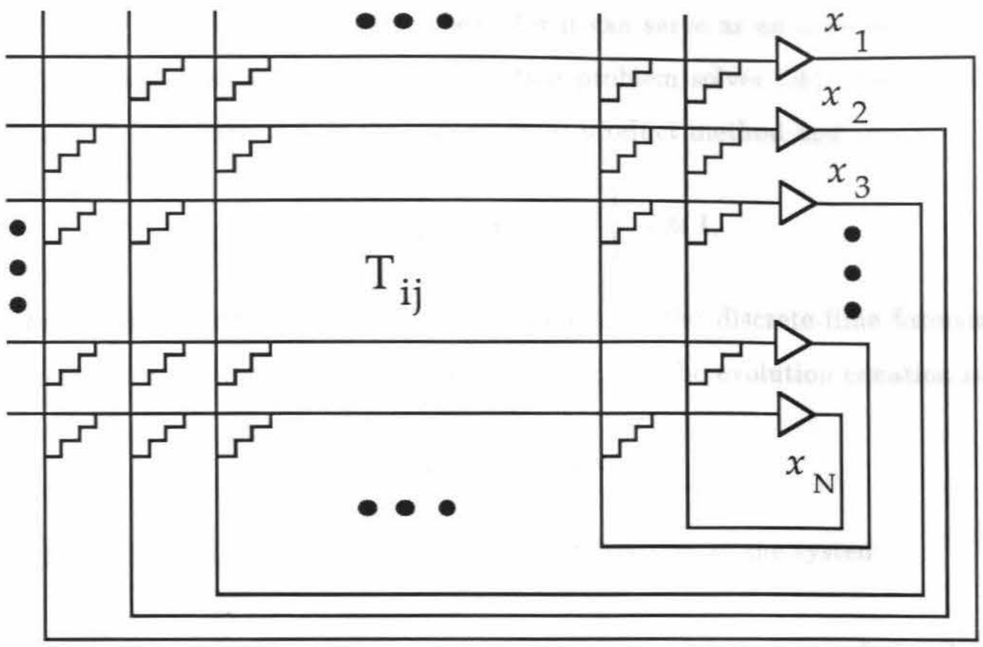


Figure 1.4: A fully-connected neural network

1.5.1 Fully-Connected Neural Network Models

This type of associative memories have N neurons, each of which is connected to the other $N-1$ neurons and possibly to itself through connection weights. The system's evolution can be expressed in a discrete-time or a continuous-time formulation. For the discrete-time formulation, the neurons can be either hard-limiter neurons or sigmoidal neurons. Furthermore, there are two possible operation modes for the discrete-time formulation : *synchronous* and *asynchronous*; the former case is when all N neurons update themselves at the same time, and the latter is when one and only one neuron updates itself in every iteration. Figure 1.4 illustrates the architecture of this type of neural network associative memories. Most models differ from the others only in how the connection weight matrix and thresholds are determined.

- **Hopfield Model**

This is by far the most popular model, for it can serve as an associative memory [24, 25] as well as a quadratic optimization problem solver [26]. The connection matrix $\mathbf{T}$ is constructed by the sum-of-outer-product method and

$$\mathbf{T} \equiv \left\{ \sum_{k=1}^{M} \mathbf{u}^{(k)} \, \mathbf{u}^{(k)^t} \right\} - M\,\mathbf{I}, \tag{1.8}$$

where $\mathbf{I}$ is the identity matrix. The system uses the discrete-time formulation, hard-limiter neurons, and asynchronous updating. The evolution equation is

$$\mathbf{x}' = sgn\,\{\mathbf{T}\,\mathbf{x}\}, \tag{1.9}$$

where $\mathbf{x}'$, $\mathbf{x}$ are the next and the current state pattern of the system.

A major handicap of the Hopfield model is its limited capacity. It has been reported that only $0.15N$ patterns can be stored in the Hopfield memory with N neurons for N between 30 and 100 [24]. McEliece *et al.* [43] also proved that as N approaches infinity, the capacity of the Hopfield model grows no faster than $N/2\log N$. Other drawbacks include : 1) N^2 connection weights are needed, and 2) the model converges to a stable state only in the asynchronous updating mode.

- **Correlation-Matrix Associative Memory**

Many researchers, through a couple of decades, have proposed a type of memory called correlation-matrix memory or nonholographic memory [4, 5, 34, 46, 69]. At first, they are intended to be used as linear associators; namely, the neurons are linear neurons without any nonlinearity. Most suggested models focused on heteroassociative memories; however, Kohonen showed that it can be also used as an autoassociative memory. The correlation-matrix associative memory constructs its connection matrix by the sum-of-outer-product rule except that, unlike the Hopfield model, the diagonal is not zeroed. Gindi *et al.* [18] and Stiles and Denq [63] have independently tried this connection matrix on a fully-connected network with hard-limiter neurons. Both groups found that the storage capacity of this associative memory is approximately $0.15N$, which is about the same as that of the original Hopfield model. In addition, it can be shown that this associative memory converges in *both* synchronous and asynchronous updating modes (see Chapter four for detail).

• **Pseudoinverse Associative Memory and the Spectral Method**

Pseudoinverses of matrices are also called generalized inverses and they possess many properties that true matrix inverses do [32, 33]. However, they are more general because a matrix can be nonsquare and it still have a pseudoinverse, while a matrix must be square in order to have a true inverse. The connection weight matrix for the pseudoinverse associative memory [32, 48] is defined as

$$\mathbf{T} \equiv \mathbf{U} \left(\mathbf{U}^t \mathbf{U}\right)^{-1} \mathbf{U}^t, \tag{1.10}$$

where the kth column of $\mathbf{U}$ is the kth memory pattern $\mathbf{u}^{(k)}$.

Venkatesh [65] generalized the pseudoinverse method and proposed a spectral method, in which the "strength" of any memory pattern can be controlled by adjusting the magnitude of the corresponding eigenvalue. Suppose that the M memory patterns are to have strengths of $\lambda_1, \lambda_2, \cdots, \lambda_M$, respectively; then the connection matrix is

$$\mathbf{T} \equiv \mathbf{U}\,\mathbf{\Lambda} \left(\mathbf{U}^t \mathbf{U}\right)^{-1} \mathbf{U}^t, \tag{1.11}$$

where $\mathbf{\Lambda}$ is a diagonal matrix with entries $\lambda_1, \lambda_2, \cdots, \lambda_M$ in the diagonal. It was also shown by simulation that the spectral method is better than the Hopfield model in terms of the storage capacity and the ability to perform perfect recall.

Poggio [51] proposed a general nonlinear associative recall method, whose first order approximation is the pseudoinverse method. However, the procedure is very complicated and we will evade further discussion.

- **Associative Memory Trained by the Backpropagation**

  Kam *et al.* [30] offered another approach to building the connection matrix for a fully-connected neural network. He proposed that the backpropagation learning rule be used to generate the connection weights. In this scheme not only the associative recall function can be performed but also regions of attraction of the memory patterns can be controlled. He also claimed that this model has larger attraction regions as well as a smaller number of spurious states than the Hopfield model.

1.5.2 Correlation Associative Memories

By its definition, associative memory should produce, as its output, the nearest memory pattern to the input pattern. The similarity measure used is often some distance : Hamming distance for binary patterns and Euclidean distance for real patterns. In the case of binary memory patterns and the case of real memory patterns with equal distances to the origin, the previous two distance measures can both be replaced by the ordinary correlation. The following binary associative memories all take advantage of this fact and perform autoassociative recall by 1) calculating the correlations between the input binary pattern and all M binary memory patterns; 2) applying certain function to those correlations; 3) using the previous quantities as weights to compute a weighted sum of all M memory patterns; 4) thresholding the result component by component; 5) feeding the output back to the input side; and 6) repeating this procedure until the system reachs a stable state, which is the final recalled pattern. The evolution equation of the correlation associative memories based on the ordinary correlation is

$$\mathbf{x}' = sgn\left\{\sum_{k=1}^{M} f(<\mathbf{u}^{(k)},\mathbf{x}>)\,\mathbf{u}^{(k)}\right\}, \tag{1.12}$$

where $<,>$ is the ordinary correlation (inner product) of two patterns. Next we will discuss existing correlation associative memories based on the ordinary correlation one by one.

- **Correlation-Matrix Associative Memory**

This aforementioned memory can be formulated as an instance of the correlation associative memories based on the ordinary correlation with the function $f(t) \equiv t$.

- **High-Order Correlation Associative Memory**

A few groups of researchers independently suggested the use of high-order correlation matrices in the implementation of associative memories [17, 38, 53, 61, 62]. In this type of memory the nonlinear function has the form of some power of the ordinary correlation

$$f(t) \equiv t^q, \tag{1.13}$$

where q is usually an integer and $q > 1$. The storage capacity of this type of associative memories has been shown to be proportional to N^q asymptotically [62], which is much larger than that of the Hopfield model.

- **Potential Function Correlation Associative Memory**

Sayeh and Han [57] and Dembo and Zeitouni [12, 13] independently proposed this model, which is based on the distance measure rather than on the correlation. The original model is for a continuous-time system and patterns with real components, yet it is easily transformed into a discrete-time formulation for binary patterns. The evolution equation is Equation (1.12) with

$$f(t) \equiv \frac{1}{(N-t)^L}, \tag{1.14}$$

where L is an integer and $L > 3$. Note that $N - < \mathbf{u}^{(k)}, \mathbf{x} >$ is proportional to the Hamming distance between $\mathbf{u}^{(k)}$ and $\mathbf{x}$. The storage capacity of this model has been demonstrated to be limited only by the information theory sphere-packing bound. The major disadvantage of this model is that hardware implementation of the nonlinear potential function is very complicated.

- **Exponential Correlation Associative Memory (ECAM)**

  We introduced a new correlation associative memory that uses the exponentiation function in Equation (1.12) and

$$f(t) \equiv a^t, \tag{1.15}$$

  where $a > 1$ [10]. The capacity of ECAM has been proved to be proportional to c^N, where the constant c depends on the required error probability and percentage error in input patterns (see Chapter four for detail). ECAM was designed with a view to implementing it by VLSI. It turns out that ECAM is most amenable to VLSI implementation because MOS transistors exhibit an exponential relationship between the drain current and the gate-to-source voltage when working in the subthreshold region.

All the aforementioned correlation associative memories can be shown to converge to a stable state in both synchronous and asynchronous updating modes (see Chapter four). Since the latter three models all seem to have larger storage capacity than needed in most real-life applications, an important factor in favoring one over the other is the ease of electronic or optical realization. For instance, the high-order associative memory with $q = 2$ is easily realized by optical components, while ECAM can be readily implemented using VLSI technology.

1.5.3 Coded Associative Memory

This type of associative memories all have the following features : 1) input patterns are transformed to vectors in a code space, and 2) for each memory pattern $\mathbf{u}^{(k)}$, there is a corresponding codeword $\mathbf{v}^{(k)}, k = 1, 2, \ldots, M$.

- **Hamming Network**

The code used in this associative memory is the unary code (grandmother cell code) [6, 39]. In this code, codewords are M bits wide, and the k^{th} codeword has all "0" bits except at the k^{th} position, which has a "1." At first, the system calculates the matching scores (correlations) of the input pattern with all M memory patterns using M hidden units. It then passes those scores to a winner-take-all circuit to determine the maximum, and forms the right codeword, which is then used to retrieve the nearest memory pattern. In principle, if the winner-take-all circuit can operate perfectly, this system can find the nearest memory pattern in one operation. Nevertheless, winner-take-all circuits usually have some defects and are not able to pick the maximum value precisely.

- **Distributed Coded Associative Memory**

This associative memory was introduced in order to reduce the number of hidden units required in Hamming network. Baum *et al.* [6] proposed an associative memory model with the following distributed code : Each codeword consists of S groups and the l^{th} group has r_l bits, where r_l's are assumed to be relative prime. For the k^{th} codeword, the j^{th} bit in the l^{th} group will be "1" if and only if

$$k \equiv j \bmod r_l,$$

and "0" otherwise. The mapping is done by a one-layer perceptron whose connection matrix is constructed by the sum of outer products of the memory patterns and the codewords. A second layer is added to map noisy codewords back to the feature pattern space. Also, a feedback loop can be augmented to make the system a feedback associative memory.

- **Kanerva Memory**

Kanerva [31] introduced an associative memory model, which he claimed to be closely related to the human cerebellar model of Marr [42] and Albus [2]. His model can be formulated as a two-layer perceptron [11]. The first connection matrix is made up of random $+1$ or -1 entries. M random codewords can be found by feeding M corresponding memory patterns to the first layer. The second layer is another perceptron whose connection weights are computed by the sum of outer products of the memory patterns and the random codewords. Again, this system can be made recurrent by making a connection between the output end and the input end.

1.6 Discussions

In the previous sections, we have given an overview of two important functions in information processing systems, i. e., pattern classification and associative recall, together with their neural network implementation. In this section, we will discuss two interesting issues concerning pattern classifiers and associative memories.

1.6.1 Relationship Between Pattern Classifiers and Associative Memories

As mentioned previously, the simplest form of classification is when all conditional probability distributions are multivariate Gaussian and when input features are uncorrelated. In this case, the general classification problem becomes the minimum distance classification problem. It can then be treated as a special case of heteroassociative memories, whose response patterns are unary codewords that encode the indices of output categories. Consequently, one can build a minimum distance classifier by storing all M prototypes as memory patterns in an autoassociative memory and cascading the autoassociative memory with a proper content-addressable lookup table.

On the other hand, an autoassociative memory can be constructed based on a minimum distance classifier in the following way : M memory patterns are used as prototypes to design a minimum distance classifier. An input pattern can be fed to that classifier and the result is a unary codeword encoding the index of the correct class. That codeword is then used as an address for a lookup table, which stores all M memory patterns, to retrieve the nearest memory pattern.

1.6.2 Winner-Take-All Circuits

"Winner-take-all" networks are supposed to identify, among a set of an undetermined yet fixed number of neurons, the neuron with the maximum activation, and to enhance its output while at the same time suppressing those of all the others. The winner-take-all function is a vital component in many neural network systems. Since they perform such an important function, many winner-take-all circuits have been suggested. In the following, we will present some existing winner-take-all circuits and discuss their disadvantages.

One realization of the winner-take-all function is a binary tree made up of two-input comparators introduced in [19, 39]. The drawbacks of this architecture include the following : 1) The number of neurons and weights needed is large, and 2) signals have to travel $\log_2 M$ levels of circuits and might have degraded to such a degree that accurate operation is not possible. The other scheme suggested by Winters and Rose [70] utilizes a cellular automata, and it also suffers from the signal degradation problem.

A fully-connected neural network with N neurons that performs the winner-take-all function has been proposed by many researchers [21, 39]. The main disadvantage of this type of implementation is the number of connection weights needed is the square of the number of categories in the system. Mjolsness and Garrett [44] and Majani, *et al.* [41] both suggested variations of the aforementioned globally inhibitory network, and they require only $O(M)$ hardware complexity, where M is the number of classes in the system . These models, though more efficient in terms of hardware complexity, are still continuous-time feedback networks and thus take an indefinite amount of time to converge.

An MOS VLSI implementation of the winner-take-all function was introduced by Lazzaro and his colleagues [36]. This circuit is essentially a generalization of two-input differential amplifiers and requires only $O(M)$ complexity. However, it demands that the maximum input be larger than all the other inputs by a significant amount, which is often not true in real applications.

References

[1] S. C. Ahalt, F. D. Garger, I. Jouny, and A. K. Krishnamurthy, "Performance of Synthetic Neural Network Classification of Noisy Radar Signals," in *Advances in Neural Information Processing Systems*, Vol. 1, Palo Alto, CA : Morgan Kaufmann Publishers, 1989, pp. 281–288.

[2] J. S. Albus, *Brain, Behavior, and Robotics.* Peterborough, NH : BYTE book of McGraw-Hill, 1981.

[3] J. A. Anderson and E. Rosenfeld (editors), *Neurocomputing — Foundation of Research.* Cambridge, MA : MIT Press, 1988.

[4] J. A. Anderson and G. L. Murphy, "Psychological Concepts on a Parallel System," *Physica 22D*, 318–336, 1986.

[5] J. A. Anderson, "Cognitive Capabilities of a Parallel System," in *Disordered Systems and Biological Organization*, E. Brenenstock (editor), Berlin : Springer-Verlag, 1986.

[6] E. B. Baum, J. Moody, and F. Wilczek, "Internal Representation for Associative Memory," *Biological Cybernetics*, Vol. 59, 217–228, 1988.

[7] S.-T. Bow, *Pattern Recognition — Application to Large Data-Set Problems.* New York, NY : Marcel Dekker, 1984.

[8] G. A. Carpenter and S. Grossberg, "ART 2 : Self-Organization of Stable Category Recognition Codes for Analog Input Patterns," *Applied Optics*, Vol. 26, 4919–4930, 1987.

[9] T. D. Chiueh and R. M. Goodman, "A neural Network Classifier Based on Coding Theory," in *Neural Information Processing Systems*, D. Z. Anderson (editor), New York, NY : American Institute of Physics, 1988, pp. 174–183.

[10] T. D. Chiueh and R. M. Goodman, "High-Capacity Exponential Associative Memory," in *Proc. Int. Conf. on Neural Networks*, San Diego, CA, Vol. I, 1988, pp. 153–160.

[11] P. A. Chou, "The Capacity of the Kanerva Associative Memory is Exponential," in *Neural Information Processing Systems*, D. Z. Anderson (editor), New York, NY : American Institute of Physics, 1988, pp. 184–191.

[12] A. Dembo and O. Zeitouni, "High Density Associative Memories," in *Neural Information Processing Systems*, D. Z. Anderson (editor), New York, NY : American Institute of Physics, 1988, pp. 211–218.

[13] A. Dembo and O. Zeitouni, "General Potential Surfaces and Neural Networks," *Physical Review A*, Vol. 37, No. 6, 2134–2143, 1988.

[14] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis.* New York, NY : John Wiley and Sons, 1973.

[15] K. Fukunaga, "Statistical Pattern Classification," in *Handbook of Pattern Recognition and Image Processing*, T. Y. Young and K.-S. Fu (editors), San Diego, CA : Academic Press, 1986.

[16] K. Fukunaga, *Introduction to Statistical Pattern Recognition.* Orlando, FL : Academic Press, 1972.

[17] C. L. Giles and T. Maxwell, "Learning, Invariance, and Generalization in High-Order Neural Networks," *Applied Optics*, Vol. 26, 4972–4978, 1987.

[18] G. R. Gindi, A. F. Gmitro, and K. Parthasarathy, "Hopfield Model Associative Memory with Nonzero-Diagonal Terms in Memory Matrix," *Applied Optics*, Vol. 27, No. 1, 129–134, 1988.

[19] G. R. Gindi, A. F. Gmitro, and K. Parthasarathy, "Winner-Take-All Networks and Associative Memory : Analysis and Optical Realization," in *Proc. Int. Conf. on Neural Networks*, San Diego, CA, Vol. III, 1987, pp. 607–614.

[20] R. M. Glorioso and F. C. Colon Osorio, *Engineering Intelligent Systems — Concepts, Theory, and Applications.* MA : Digital Press, 1980.

[21] S. Grossberg, "Nonlinear Neural Networks : Principles, Mechanisms, and Architectures," *Neural Networks*, Vol. 1, 17–61, 1988.

[22] J. A. Hartigan, *Clustering Algorithms*, New York, NY : Wiley and Sons, 1975.

[23] Y.-C. Ho and R. L. Kashyap, "A Class of Iterative Procedure for Linear Inequalities," *Journal of SIAM Control*, Vol. 4, 112–115, 1966.

[24] J. J. Hopfield, "Neural Network and Physical Systems with Emergent Collective Computational Abilities," *Proc. Nat. Acad. Sci. USA*, Vol. 79, 2554–2558, 1982.

[25] J. J. Hopfield, "Neurons with Graded Response Have Collective Computational Properties Like Those of Two-State Neurons," *Proc. Nat. Acad. Sci. USA*, Vol. 81, 3088–3092, 1984.

[26] J. J. Hopfield and D. W. Tank, "Neural Computation of Decisions in optimization Problems," *Biological Cybernetics*, Vol. 52, 141–152, 1985.

[27] W. Y. Huang and R. P. Lippmann, "Neural Net and Traditional Classifiers," in *Neural Information Processing Systems*, D. Z. Anderson (editor), New York, NY : American Institute of Physics, 1988, pp. 387–396.

[28] W. Y. Huang and R. P. Lippmann, "Comparisons Between Neural Net and Conventional Classifiers," in *Proc. Int. Conf. on Neural Networks*, San Diego, CA, Vol. IV, 1987, pp. 485–494.

[29] A. K. Jain, "Cluster Analysis," in *Handbook of Pattern Recognition and Image Processing*, T. Y. Young and K.-S. Fu (editors), San Diego, CA : Academic Press, 1986.

[30] M. Kam, R. Cheng, and A. Guez, "On the Design of a Content-Addressable Memory," in *Proc. Int. Conf. on Neural Networks*, San Diego, CA, Vol. II, 1987, pp. 513–522.

[31] P. Kanerva, "Parallel Structure in Human and Computer Memory," in *Neural Networks for Computing*, J. S. Denker (editor), New York, NY : American Institute of Physics, 1986, pp. 247–258.

[32] T. Kohonen, *Associative Memory : A System Theoretic Approach.* Berlin : Springer-Verlag, 1977.

[33] T. Kohonen, *Self Organization and Associative Memory*, 2nd edition. Berlin : Springer-Verlag, 1988.

[34] T. Kohonen, "Correlation Matrix Memory," in *Neurocomputing — Foundation of Research*, J. A. Anderson and E. Rosenfeld (editors), Cambridge, MA : MIT Press, 1988.

[35] A. Lapedes and R. Farber, "How Neural Networks Work," in *Neural Information Processing Systems*, D. Z. Anderson (editor), New York, NY : American Institute of Physics, 1988, pp. 442–456.

[36] J. P. Lazzaro, S. Ryckebusch, M. A. Mahowald, and C. A. Mead, " Winner-Take-All Networks of $O(N)$ Complexity," in *Advances in Neural Information Processing Systems*, Vol. 1, Palo Alto, CA : Morgan Kaufmann Publishers, 1989, pp. 703-711.

[37] K. Lee, "Neural Network Applications in Handwritten Symbol Understanding," Presented at the INNS First Annual Meeting, Boston, MA, 1988.

[38] Y. C. Lee, G. Doolen, H. H. Chen, G. Z. Sun, T. Maxwell, H. Y. Lee, and C. Lee Giles, "Machine Learning Using a Higher Order Correlation Network," *Physica 22D*, 276–306, 1986.

[39] R. Lippmann, "An Introduction to Computing with Neural Nets," *IEEE ASSP Magazine*, Vol. 4, No. 2, 4–22, 1987.

[40] R. P. Lippmann and B. Gold, "Neural Net Classifier Useful for Speech Recognition," in *Proc. Int. Conf. on Neural Networks*, San Diego, CA, Vol. IV, 1987, pp. 417–426.

[41] E. Majani, R. Erlanson, and Y. Abu-Mostafa, " On the K-Winner-Take-All Network," in *Advances in Neural Information Processing Systems*, Vol. 1, Palo Alto, CA : Morgan Kaufmann Publishers, 1989, pp. 634–642.

[42] D. Marr, "A Theory of Cerebellar Cortex," *Journal of Physiology*, Vol. 202, 437–470, 1969.

[43] R. J. McEliece, E. C. Posner, E. R. Rodemich, S. S. Venkatesh, "The Capacity of The Hopfield Associative Memory," *IEEE Trans. on Information Theory*, Vol. IT-33, 461–482, 1987.

[44] E. Mjolsness and C. Garrett, "Algebraic Transformations of Objective Functions," Research Report, Department of Computer Science, Yale University, YaleU/DCS/RR-686, 1989.

[45] B. Moor, "ART 1 and Pattern Clustering," in *Proc. of Connectionist Summer School 1988*, Palo Alto, CA : Morgan Kaufmann Publishers, 1988.

[46] K. Nakano, "Associatron — A Model of Associative Memory," *IEEE Trans. on System, Man, and Cybernetics*, Vol. SMC-2, 380–388, 1972.

[47] N. J. Nilsson, *Learning Machines : Foundations of Trainable Pattern Classifying Systems.* New York , NY : McGraw-Hill, 1965.

[48] G. Palm, "On Associative Memory," *Biological Cybernetics*, Vol. 36, 19–31, 1980.

[49] E. A. Patrick, *Fundamentals of Pattern Recognition.* Englewood Cliffs, NJ : Prentice-Hall, 1972.

[50] T. F. Pawlicki, D. Lee, J. J. Hull, and S. N. Srihari, "Neural Network Models and Their Application to Handwritten Digit Recognition," in *Proc. Int. Conf. on Neural Networks*, San Diego, CA, Vol. II, 1988, pp. 63–70.

[51] T. Poggio, "On Optimal Nonlinear Associative Recall," *Biological Cybernetics*, Vol. 19, 201–209, 1975.

[52] E. C. Posner, "Potential Neural Network Applications," in *Proc. of Workshop on Neural Network Devices and Applications*, Jet Propulsion Lab., Feb., 1987, pp. 1–11.

[53] D. Psaltis and C. H. Park, "Nonlinear Discriminant Functions and Associative Memory," in *Neural Networks for Computing*, J. S. Denker (editor), New York, NY : American Institute of Physics, 1986, pp. 370–375.

[54] D. L. Reilly, C. Scofield, C. Elbaum, and L. N. Cooper, "Learning System Architectures Composed of Multiple Learning Modules," in *Proc. Int. Conf. on Neural Networks*, San Diego, CA, Vol. II, 1987, pp. 495–504.

[55] F. Rosenblatt, *Principles of Neurodynamics : Perceptrons and the Theory of Brain Mechanism.* Washington, DC : Spartan Books, 1962.

[56] D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing, Explorations in the Microstructure of Cognition*, Vol. I, Cambridge, MA : MIT Press, 1987.

[57] M. R. Sayeh and J. Y. Han, "Pattern Recognition Using a Neural Network," in *Proc. of SPIE Cambridge Symp. on Opt. and Optoelec. Eng.*, Cambridge, MA, Nov., 1987.

[58] C. L. Scofield, D. L. Reilly, C. Elbaum, and L. N. Cooper, "Pattern Class Degeneracy in an Unrestricted Storage Density Memory," in *Neural Information Processing Systems*, D. Z. Anderson (editor), New York, NY : American Institute of Physics, 1988, pp. 674–682.

[59] G. S. Sebestyen, *Decision-Making Processes in Pattern Recognition.* New York, NY : Macmillan, 1962.

[60] T. J. Sejnowski and C. M. Rosenberg, "Parallel Networks that Learn to Pronounce English Text," *Complex Systems*, Vol. 1, 145–168, 1987.

[61] A. Shiozaki, "Recollection Ability of Three-Dimensional Correlation Matrix Associative Memory," *Biological Cybernetics*, Vol. 54, 337–342, 1984.

[62] B. Soffer, "Holographic Associative Memories," in *Proc. of Workshop on Neural Network Devices and Applications*, Jet Propulsion Lab., Feb., 1987, pp. 125–146.

[63] G. S. Stiles and D.-L. Denq, "A Quantitative Comparison of the Performance of Three Discrete Distributed Associative Memory Models," *IEEE Trans. on Computers*, Vol. C-36, No. 3, 257–263, 1987.

[64] J. T. Tou and R. C. Gonzalez, *Pattern Recognition Principles.* Reading, MA : Addison-Wesley, 1974.

[65] S. S. Venkatesh, "Linear Map with Point Rules," Ph. D. dissertation, California Institute of Technology, 1987.

[66] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang, "Phoneme Recognition Using Time-Delay Neural Networks," Technical Report, ATR Interpreting Telephony Research Laboratories, TR-I-0006, 1987.

[67] B. Widrow and M. E. Hoff, "Adaptive Switching Circuits," in *Neurocomputing — Foundation of Research*, J. A. Anderson and E. Rosenfeld (editors), Cambridge, MA : MIT Press, 1988.

[68] A. Wieland, R. Leighton, and G. Jacyna, "An Analysis of Noise Tolerance for a Neural Network Recognition System," Technical Report, MITRE, MP-88W00021, 1988.

[69] D. J. Willshaw, O. P. Buneman, and H. C. Longuet-Higgins, "Non-Holographic Associative Memory," *Nature*, Vol. 222, 960–962, 1969.

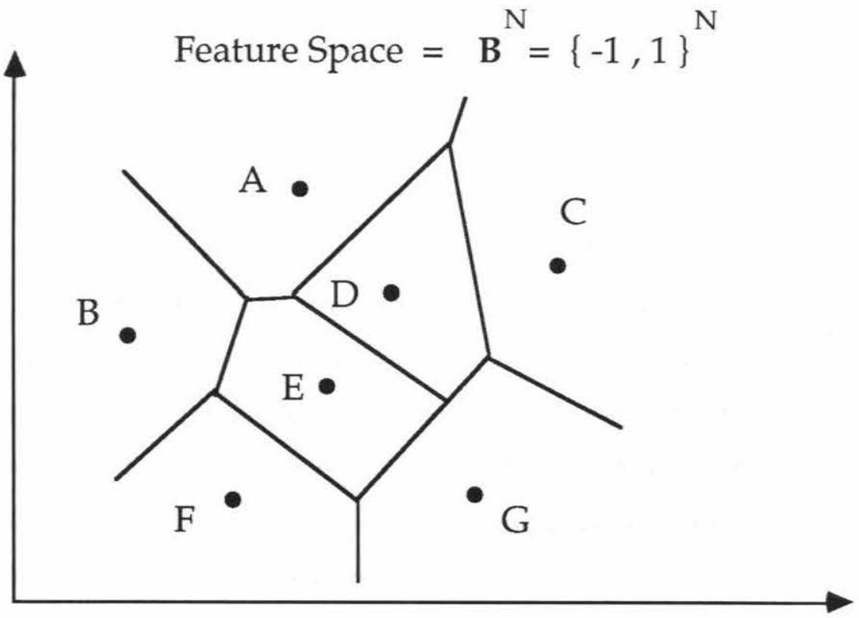
[70] J. H. Winters and Christopher Rose, "On Parallel Networks for Optimum Classification," Presented at the INNS First Annual Meeting, Boston, MA, 1988, submitted to *Neural Networks.*

Chapter 2

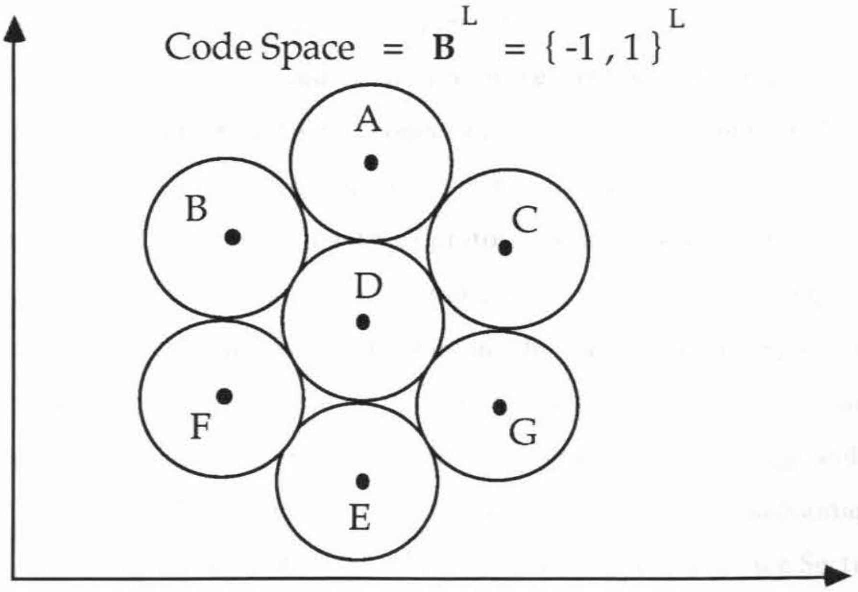
A Two-Layer Feed-Forward Network Classifier Based on Coding Theory

2.1 Introduction

In this chapter, we are specifically concerned with binary pattern classification problems. As mentioned in Chapter one, if the conditional probability distributions of all classes are assumed to be multivariate Gaussian and if all N features in input patterns are uncorrelated, then the optimal Bayes decision rule becomes finding the prototype with minimum Euclidean distance to the input pattern. Moreover, if input features are binary-valued, the minimum Euclidean distance classification problem becomes the minimum Hamming distance classification problem. In this chapter we propose a new neural network classifier for solving the minimum Hamming distance classification problem, based on the established technique of error control coding. Consider a typical minimum Hamming distance classification problem (see Figure 2.1(a)). In this problem, one is given *a priori* a set of classes, $C^{(k)}, k = 1, 2, \ldots, M$, together with their corresponding prototypes, i. e., patterns that are most representative of their classes. The output of the classifier under consideration is the index of the class whose prototype is nearest the input in Hamming distance. Therefore, the N-dimensional binary feature space



(a)



(b)

Figure 2.1: (a) The minimum Hamming distance classification problem vs. (b) the error correction decoding problem

$\mathbf{B}^N \equiv \{1, -1\}^N$ is partitioned into M decision regions, one for each class. All patterns inside each region are to be categorized to the corresponding class by the classifier.

A similar problem is that of decoding noisy codewords by an error control code decoder as shown in Figure 2.1(b). In this case codewords are constructed by design and are usually at least d_{min} bits apart. The received corrupted codeword is the input to the decoder, which finds the nearest codeword to the input in Hamming distance. In principle, if the Hamming distances between all pairs of codewords are greater than $2b + 1$, it is possible to decode (classify) a noisy codeword (binary pattern) and then find the correct codeword (prototype) provided that the Hamming distance between the noisy codeword and the nearest codeword is no more than b. However, in the minimum Hamming distance classification problem, there is no guarantee that the prototypes are uniformly distributed in $\mathbf{B}^N$; consequently, the attraction radius (the maximum number of errors that can occur in any given input pattern such that it can still be correctly classified) depends on the *minimum* Hamming distance among all prototypes.

Many solutions to the minimum Hamming distance classification problem have been suggested. The one commonly used is the Hamming network, which is similar to the matched filter construction in statistical communication theory. Lippmann [12] proposed a two-stage Hamming network that solves the minimum distance classification problem by first correlating the input pattern with all prototypes, and obtaining the class index of the nearest prototype by picking the maximum correlation. Lippmann suggested using a "winner-take-all" circuit to pick the maximum. In Figure 2.2, $x_1, x_2, \cdots, x_N$ are N binary input features, and $y_1, y_2, \cdots, y_M$ are the correlations (similarity measures) of $\mathbf{x}$ with M prototypes. The second block picks the maximum of $y_1, y_2, \cdots, y_M$ and produces the index of the class with maximum similarity measure. The main disadvantage of such a classifier is the constraints on the winner-take-all circuit (for detail, see Section 1.6.2). Recently, Lazzaro *et al.* [11] and Majani *et al.* [13] proposed new architectures for the winner-take-all function. However, both architectures have some drawbacks : Lazzaro's architecture requires that the maximum input value be much larger than the others,

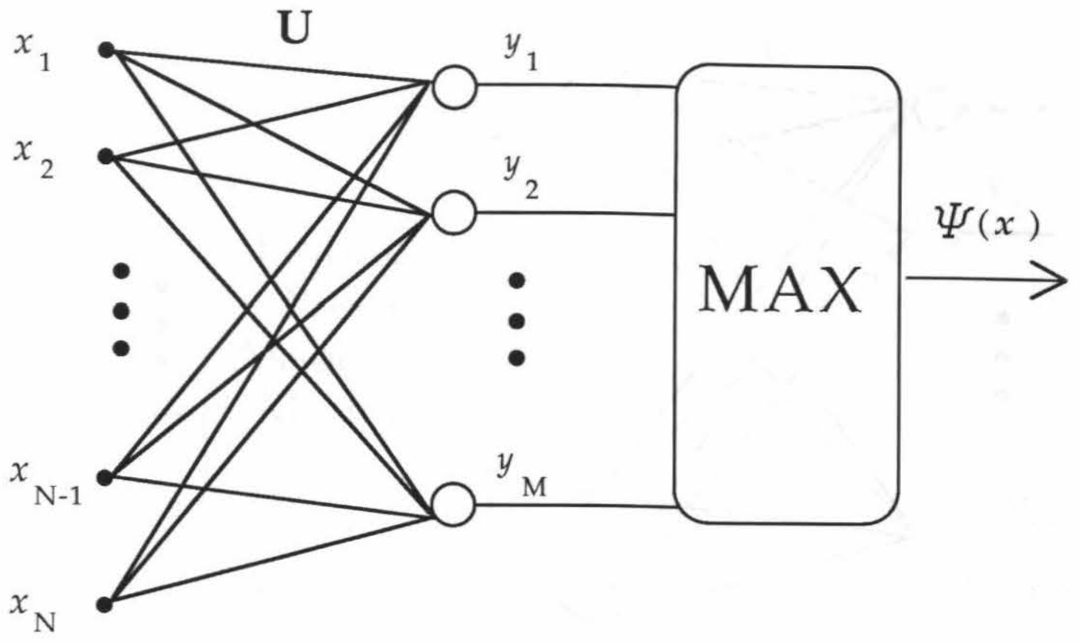


Figure 2.2: A matched filter type classifier

which is often not true. Majani's circuit uses a continuous-time Hopfield memory, which might take a long time to converge.

A second solution is to use feed-forward networks with hidden units and to apply the backpropagation rule (generalized delta rule) [16]. However, it has been reported that applying the backpropagation rule to training sets on networks with hidden units is usually a slow process. What's worse, in minimum Hamming distance classification problems, training sets are so huge that the time it takes to train is usually unbearably long.

Another alternative to solving the minimum Hamming distance classification problem is to build a classifier based on the Hopfield memory [6, 7]. This design consists of a fully-connected feedback network and a correlation-calculating connection matrix

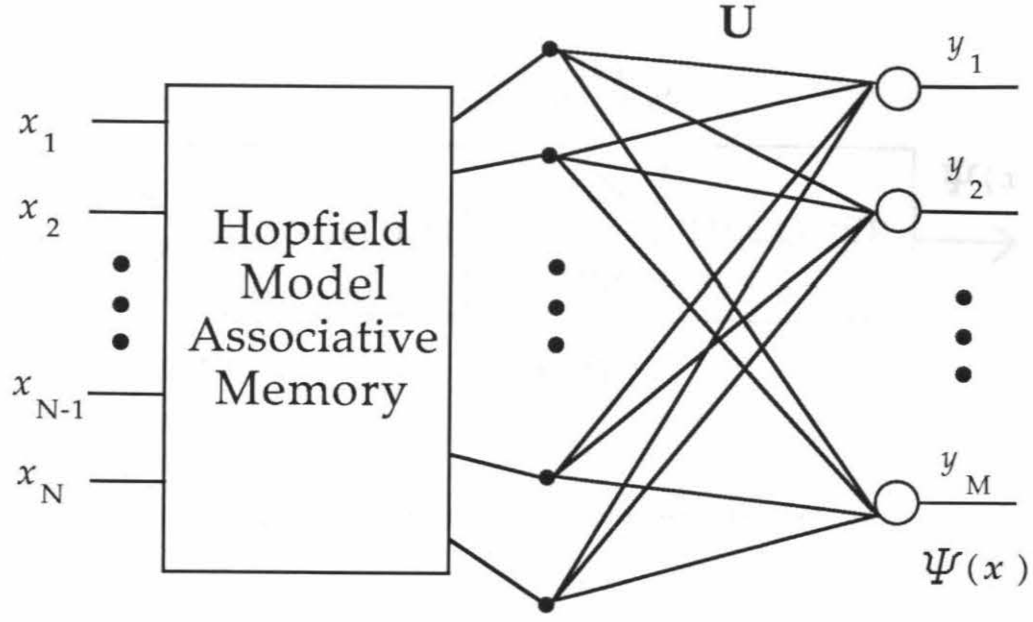


Figure 2.3: A classifier based on the Hopfield memory

followed by M hard-limiter neurons with thresholds all set at $N - 1/2$ (see Figure 2.3). At first the Hopfield memory is programmed with M prototypes as memory patterns. The input pattern is fed to the Hopfield memory, and the Hopfield memory is allowed to run until it becomes stable. Then the output of the Hopfield memory is checked against all M prototypes $\mathbf{u}^{(k)}$, $k = 1, 2, \ldots, M$ by the second stage. The output neuron whose prototype is identical to the response of the Hopfield memory will be activated $(+1)$. If the response of the Hopfield memory is not any of the M prototypes, all output neurons will be off (-1), which signifies rejection of the input pattern.

It has been reported that the number of memory patterns that can be stored in an N-neuron Hopfield memory is about $0.15N$ for N between 30 and 100 [6]. McEliece *et al.* showed that in synchronous updating mode, the Hopfield memory stores about $N/2\log N$ memory patterns reliably when N approaches infinity [14]. Abu-Mostafa and

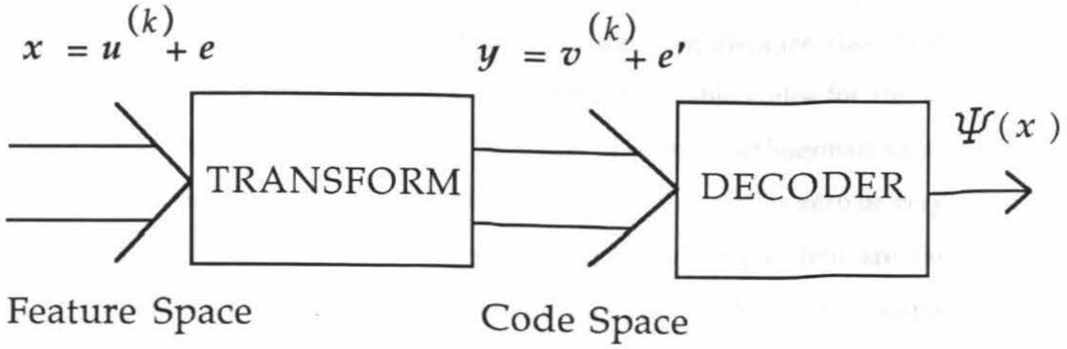


Figure 2.4: Structure of the proposed classifier

St. Jacques [1] predicted that the upper bound for the number of memory patterns that can be stored in an N-neuron fully-connected feedback network is N. We believe that it is possible to design a new classifier with M — the number of stored prototypes, linear in N for large N.

Our main idea is to map patterns in the feature space to vectors in some code space so that each prototype corresponds to a codeword in that code. The code should preferably (but not necessarily) have the property that codewords are uniformly distributed in the code space; namely, the Hamming distances between all pairs of codewords are the same. With this mapping, we turn a minimum Hamming distance classification problem into a decoding problem. We then do error correction decoding on the vector in the code space to obtain the index of the nearest codeword and thus classify the original input pattern, as shown in Figure 2.4.

This chapter develops the construction of such a classifier as follows. At first we consider the problem of mapping input patterns from the feature space to the code space. Two perceptrons [15] working as heteroassociative memories are introduced for doing this mapping. The first perceptron constructs its connection weight matrix by summing outer products of the prototypes and the codewords. The second perceptron generates its connection weight matrix by the pseudoinverse technique [9]. Given that we have transformed the problem of minimum Hamming distance classification into the problem of decoding a noisy codeword, we consider suitable codes for the new classifier. The codewords in this code should be orthogonal or pseudo-orthogonal; namely, the ratio of the cross-correlation to the autocorrelation of the codewords is zero or very small. Two classes of good codes suitable for this particular decoding problem are the Hadamard matrix codes and the maximal-length sequence codes [2]. Next the complete decoding algorithm is formulated, and it is shown how the new classifier can be implemented by a two-layer neural network. The first layer performs the mapping on the input pattern, and the second one decodes the corresponding vector in the code space and produces the index of the class to which the input belongs. Finally, we work a small example by hand to give a feel of the classification process.

The second part of this chapter deals with the performance of the new classifier. We first analyze the performance of the new classifier by finding the relation between the maximum number of classes that can be stored and the misclassification rate. We show (when using a mapping based on the sum-of-outer-product method) that for a negligible misclassification rate and large N, a not very tight lower bound on M is $0.22N$. Then comprehensive simulation results, which confirm and exceed our theoretical predictions when N is moderately large, are presented. The simulation results compare the new classifier with the Hopfield-memory-based classifier for both the sum-of-outer-product method and the pseudoinverse method, and for both analog and clipped connection matrices. The misclassification rate of all classifiers is set at 0.2%; in other words, if we average over many different sets of randomly chosen prototypes, the classifiers must classify correctly more than 99.8% of the time when presented with random inputs of a given error rate. In all cases the new classifier outperforms the Hopfield-memory-based classifier in terms of the number of prototypes that can be reliably stored. For example, consider the case of $N = 127$, a clipped connection matrix, and an attraction radius of zero (no error in input patterns); the Hopfield-memory-based classifier has a storage capacity of approximately 7, while the new classifier can store 83 prototypes.

2.2 Transform Techniques

Our objective is to build a classifier that discriminates among binary input patterns and classify them to the appropriate classes. Suppose $\mathbf{u}^{(k)} \in \mathbf{B}^N$ is the prototype of the corresponding class $C^{(k)}, k = 1, 2, \ldots, M$. Given the binary input pattern $\mathbf{x}$, we want the classifier to identify the class whose prototype is closest to $\mathbf{x}$ in Hamming distance; i. e., we want to calculate the classification function $\Psi, \Psi : \mathbf{B}^N \to \{1, 2, \ldots, M\}$ and

$$\begin{aligned}\Psi(\mathbf{x}) \;\equiv\; l \qquad\qquad & \text{iff } d_{\text{Hamming}}(\mathbf{u}^{(l)}, \mathbf{x}) \;<\; d_{\text{Hamming}}(\mathbf{u}^{(k)}, \mathbf{x}), \\ & k = 1, 2, \ldots, M,\; k \neq l. \qquad (2.1)\end{aligned}$$

We approach the problem by seeking a transform $\mathcal{T}$ that maps each prototype $\mathbf{u}^{(k)}$ in $\mathbf{B}^N$ to a corresponding codeword $\mathbf{v}^{(k)}$ in $\mathbf{B}^L$. An input pattern $\mathbf{x} = \mathbf{u}^{(k)} + \mathbf{e}$ is mapped to a noisy codeword $\mathbf{y} = \mathbf{v}^{(k)} + \mathbf{e}'$, where $\mathbf{e}$ is the error added to the prototype, and $\mathbf{e}'$ is the corresponding error pattern in the code space. We then do error correction decoding on $\mathbf{y}$ to get the index of the nearest codeword. Note that $\mathbf{e}'$ may not have the same Hamming weight as $\mathbf{e}$ since the mapping $\mathcal{T}$ may either generate or eliminate errors.

We require $\mathcal{T}$ to satisfy the following equation,

$$\mathcal{T}\,\mathbf{u}^{(k)} \;=\; \mathbf{v}^{(k)} \qquad\qquad k = 1, 2, \ldots, M. \qquad (2.2)$$

Two schemes for constructing $\mathcal{T}$ are proposed. Both of them are a one-layer neural network working as a heteroassociative memory. Essentially, we construct a connection weight matrix according to $\mathbf{u}^{(k)}$'s and $\mathbf{v}^{(k)}$'s, call it $\mathbf{T}$, and then define $\mathcal{T}$ as

$$\mathcal{T} \;\equiv\; sgn \circ \mathbf{T},$$

where *sgn* is the vector threshold operator that maps a vector in $\mathbf{R}^L$ to $\mathbf{B}^L$, and $\mathbf{R}$ is the field of real numbers.

Let $\mathbf{U}$ be an $N \times M$ matrix whose k^{th} column is $\mathbf{u}^{(k)}$ and $\mathbf{V}$ be an $L \times M$ matrix whose k^{th} column is $\mathbf{v}^{(k)}$. The two methods of constructing the matrix $\mathbf{T}$ are as follows:

- **Sum-of-Outer-Product Method** [9, 10] :

  In this scheme the matrix $\mathbf{T}^{(\alpha)}$ is defined as the sum of outer products of all prototypes and codewords; i. e.,

$$T_{ij}^{(\alpha)} \equiv \sum_{k=1}^{M} v_i^{(k)} \, u_j^{(k)},$$

  or equivalently,

$$\mathbf{T}^{(\alpha)} \equiv \mathbf{V}\,\mathbf{U}^t. \tag{2.3}$$

- **Pseudoinverse Method** [9] :

  We want to find a matrix $\mathbf{T}^{(\beta)}$ satisfying the following equation,

$$\mathbf{T}^{(\beta)}\,\mathbf{U} = \mathbf{V}.$$

  In general, $\mathbf{U}$ is not a square matrix, so $\mathbf{U}^{-1}$ may not exist. To circumvent this difficulty, we can find the pseudoinverse (denoted as $\mathbf{U}^\dagger$) of the matrix $\mathbf{U}$ instead. Let

$$\mathbf{U}^\dagger \equiv \left(\mathbf{U}^t\,\mathbf{U}\right)^{-1} \mathbf{U}^t; \tag{2.4}$$

  then $\mathbf{T}^{(\beta)}$ is defined as

$$\mathbf{T}^{(\beta)} \equiv \mathbf{V}\,\mathbf{U}^\dagger = \mathbf{V}\left(\mathbf{U}^t\,\mathbf{U}\right)^{-1} \mathbf{U}^t. \tag{2.5}$$

2.3 Codes

The codes we are looking for should preferably have the property that its codewords are distributed uniformly in $\mathbf{B}^L$; in other words, the distance between each pair of codewords should be the same and as large as possible. Two such families of codes are the Hadamard matrix codes and the maximal-length sequence codes. First, let us define the word *pseudo-orthogonal.*

Definition : Let $\mathbf{v}^{(k)} = (v_1^{(k)}, v_2^{(k)}, \cdots, v_L^{(k)}) \in \mathbf{B}^L$ be the k^{th} codeword of a code, where $k = 1, 2, \ldots, M$. This code is said to be *pseudo-orthogonal* if and only if

$$\begin{aligned} <\mathbf{v}^{(k)}, \mathbf{v}^{(l)}> &= \sum_{i=1}^{L} v_i^{(k)}\, v_i^{(l)} \\ &= \begin{cases} L & k = l \\ \epsilon & k \neq l \end{cases} \qquad\qquad k, l = 1, 2, \ldots, M. \end{aligned}$$

- **Hadamard Matrix Codes :**

  These are *orthogonal* codes of length L with L rows of an $L \times L$ Hadamard matrix as codewords. For these codes, $\epsilon = 0$ and the distance between any two codewords is $L/2$. It is conjectured that there exist such codes for all L that are multiples of four, thus providing a large family of codes [2].

- **Maximal-Length Sequence Codes :**

  There exists a family of maximal-length sequences (also called pseudo-random or PN sequences), generated by shift registers and a modulo-2 adder [5]. Suppose that $\phi(z)$ is a primitive polynomial of degree d over $GF(2)$ and let $L = 2^d - 1$; then if

$$\frac{1}{\phi(z)} = \sum_{i=0}^{\infty} c_i\, z^i,$$

$c_0, c_1, \ldots\ldots$ is a periodic sequence of period L (since $\phi(z) \mid z^L - 1$).

The code whose L codewords are L cyclic-shifted versions of

$$\mathbf{v} = (1 - 2c_0, 1 - 2c_1, \cdots, 1 - 2c_{L-1})$$

satisfies pseudo-orthogonality with $\epsilon = -1$. It is also apparent that the minimum distance of this code is $(L-1)/2$, which gives a correcting power of approximately $L/4$ errors for large L.

2.4 Overall Classifier Structure

Now let us describe the overall classifier structure. Essentially, it consists of the mapping $\mathcal{T}$ followed by a error correction decoder for the Hadamard matrix code or the maximal-length sequence code. Since we would like to implement the decoder by a perceptron, it is most desirable to use *threshold decoding*, i. e., decoding algorithms that work by performing bitwise modulo-2 additions, counting ones, and thresholding counts. The decoder we propose operates by first correlating the transformed pattern in the code space with all codewords and then thresholding the results at $(L+\epsilon)/2$, respectively. The rationale of this algorithm is as follows: Since the distance between every two codewords in this code is exactly $(L-\epsilon)/2$ bits, the decoder should be able to correct any error pattern with no more than $\lfloor (L-\epsilon)/4 \rfloor$ errors if the threshold is set halfway between L and ϵ (i. e., $(L+\epsilon)/2$), where $\lfloor\ \rfloor$ is the integer floor function.

Suppose the input to the decoder is $\mathbf{y} = \mathbf{v}^{(l)} + \mathbf{e}$ and $\mathbf{e}$ is of Hamming weight h; i. e., $\mathbf{e}$ has h nonzero components, we then have

$$\begin{aligned} <\mathbf{v}^{(l)}, \mathbf{y}> &= L - 2h \\ <\mathbf{v}^{(k)}, \mathbf{y}> &= 2h + \epsilon \qquad k = 1, 2, \ldots, M,\ k \neq l. \end{aligned}$$

From the above equation, one sees that if h is no more than $\lfloor (L-\epsilon)/4 \rfloor$, then $<\mathbf{v}^{(l)}, \mathbf{y}>$ will be no less than $(L+\epsilon)/2$ and for all $k = 1, 2, \ldots, M,\ k \neq l,\ <\mathbf{v}^{(k)}, \mathbf{y}>$ will be less than $(L+\epsilon)/2$. As a result of this observation, we arrive at the following decoding algorithm,

$$\mathcal{V}(\mathbf{y}) \equiv sgn\left(\mathbf{V}^t\mathbf{y} - \frac{L+\epsilon}{2}\mathbf{j}\right), \tag{2.6}$$

where $\mathbf{j} = (1\,1\,\cdots\,1)^t$ is an $M \times 1$ vector.

In the case when $\epsilon = -1$ and less than $\lfloor (L+1)/4 \rfloor$ errors in the decoder input, the output pattern will have only one positive component $(+1)$, the index of which is the index of the class that the input pattern is classified to. However, if there are more than $\lfloor (L+1)/4 \rfloor$ errors, the output may be either the all-negative (-1) pattern (decoder failure) or another pattern with one positive $(+1)$ component (decoder error).

The classification function can now be defined as the composition of $\mathcal{T}$ and $\mathcal{V}$,

$$\Psi \equiv \mathcal{V} \circ \mathcal{T}. \tag{2.7}$$

The overall structure of the new classifier is depicted in Figure 2.5. It can be viewed as a two-layer feed-forward neural network (perceptron) with L hidden neurons and M output neurons. Also note that all neurons are hard-limiter neurons. The first layer maps the input pattern to a noisy codeword in the code space (the "internal representation") and the second layer decodes the first's output and produces the index of the class to which the input belongs.

2.5 An Example Using the Hadamard Matrix Code

We now give an example of the classification procedure of the new classifier. Suppose $N = L = 4$ and $M = 3$; let $\mathbf{u}^{(1)} = (1\ -1\ -1\ -1)^t$, $\mathbf{u}^{(2)} = (1\ -1\ \ 1\ \ 1)^t$ and $\mathbf{u}^{(3)} = (1\ \ 1\ \ 1\ \ 1)^t$; also choose the codewords as rows in a 4×4 Hadamard matrix, so $\mathbf{v}^{(1)} = (1\ -1\ \ 1\ -1)^t$, $\mathbf{v}^{(2)} = (1\ \ 1\ -1\ -1)^t$ and $\mathbf{v}^{(3)} = (1\ -1\ -1\ \ 1)^t$ and $\epsilon = 0$. We then calculate three outer product matrices as follows:

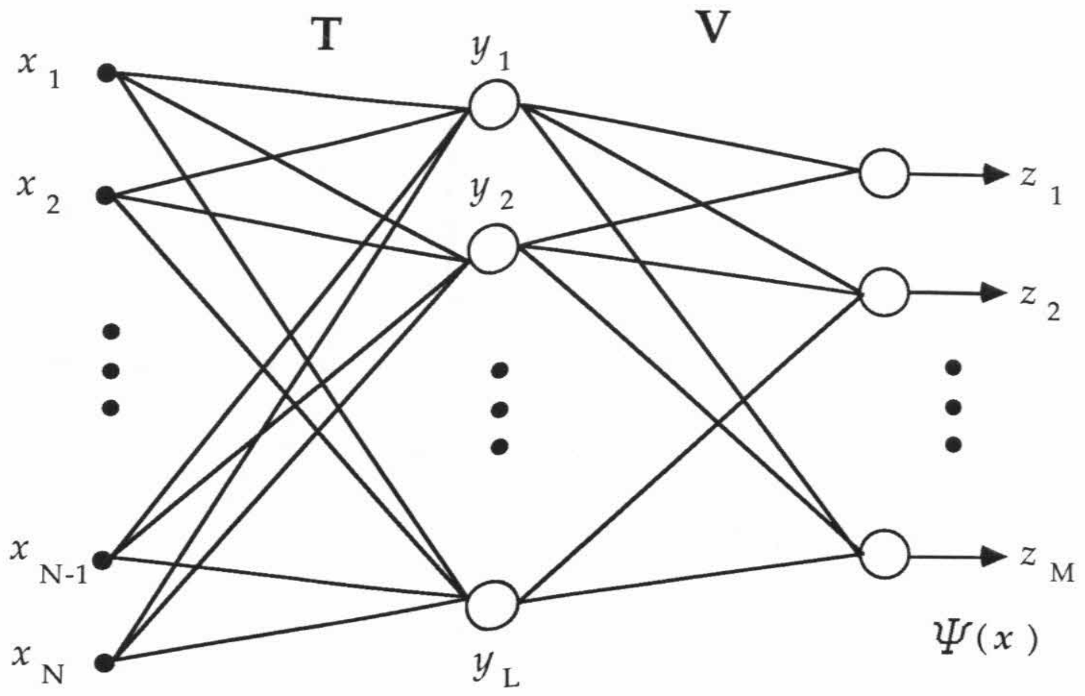


Figure 2.5: Overall architecture of the new neural network classifier

$$\mathbf{T}^{(1)} = \mathbf{v}^{(1)}\mathbf{u}^{(1)^t} = \begin{pmatrix} 1 & -1 & -1 & -1 \\ -1 & 1 & 1 & 1 \\ 1 & -1 & -1 & -1 \\ -1 & 1 & 1 & 1 \end{pmatrix}, \quad \mathbf{T}^{(2)} = \mathbf{v}^{(2)}\mathbf{u}^{(2)^t} = \begin{pmatrix} 1 & -1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ -1 & 1 & -1 & -1 \\ -1 & 1 & -1 & -1 \end{pmatrix}$$

$$\mathbf{T}^{(3)} = \mathbf{v}^{(3)}\mathbf{u}^{(3)^t} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 \end{pmatrix}.$$

Consequently, we have

$$\begin{aligned}\mathbf{T} &= \mathbf{T}^{(1)} + \mathbf{T}^{(2)} + \mathbf{T}^{(3)} \\ &= \begin{pmatrix} 3 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & -3 & -3 \\ -1 & 3 & 1 & 1 \end{pmatrix}.\end{aligned}$$

If the input pattern is $\mathbf{x} = \mathbf{u}^{(1)} = (1\ -1\ -1\ -1)^t$, then

$$\begin{aligned}\mathcal{T}(\mathbf{x}) \equiv \mathbf{y} &= sgn\,(\mathbf{T} \cdot \mathbf{x}) \\ &= sgn\left((2\ -2\ \ 6\ -6)^t\right) \\ &= (1\ -1\ \ 1\ -1)^t\end{aligned}$$

and

$$\begin{aligned}\mathbf{V}^t \cdot \mathbf{y} - \left(\frac{N}{2}\right)\mathbf{j} &= \begin{pmatrix} 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}\begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \end{pmatrix} - (2\ 2\ 2)^t \\ &= (2\ -2\ -2)^t.\end{aligned}$$

And the class index can be found by thresholding the above pattern,

$$\begin{aligned}\Psi(\mathbf{x}) &= sgn\left(\mathbf{V}^t \cdot \mathbf{y} - \left(\frac{N}{2}\right)\mathbf{j}\right) \\ &= (1\ -1\ -1)^t.\end{aligned}$$

We then conclude that the input feature vector $(1\ -1\ -1\ -1)^t$ should be classified to the first class.

2.6 Performance Analysis

From the previous sections, we know that the new classifier will make an error only if the transformed vector in the code space has more than $\lfloor (L-\epsilon)/4 \rfloor$ errors. Let us now find the error rate of the new classifier with the sum-of-outer-product construction scheme when the input is one of the prototypes, say $\mathbf{x} = \mathbf{u}^{(l)}$. Following the approach taken by McEliece *et al.* [14], we have

$$\begin{aligned}
(\mathcal{T}\,\mathbf{x})_i &= sgn\left(\sum_{j=1}^{N}\sum_{k=1}^{M} v_i^{(k)}\,u_j^{(k)}\,u_j^{(l)}\right) \\
&= sgn\left(N\,v_i^{(l)} + \sum_{j=1}^{N}\sum_{k=1,k\neq l}^{M} v_i^{(k)}\,u_j^{(k)}\,u_j^{(l)}\right).
\end{aligned}$$

Assume, without loss of generality, that $v_i^{(l)} = -1$; then

$$\left(\mathcal{T}\,\mathbf{u}^{(l)}\right)_i \neq v_i^{(l)} \qquad \text{iff} \qquad \sum_{j=1}^{N}\sum_{k=1,k\neq l}^{M} v_i^{(k)}\,u_j^{(k)}\,u_j^{(l)} \geq N. \tag{2.8}$$

Note that we assumed all $\mathbf{u}^{(k)}$'s to be random; namely, that each component of any $\mathbf{u}^{(k)}$ is the outcome of a Bernoulli trial. Accordingly, the left hand side of the inequality in Equation (2.8) is the sum of $N(M-1)$ independent, identically-distributed random variables with zero mean and unit variance. In the asymptotic case, when N and M are both very large and if the ratio of N over M stays fixed, the central limit theorem [4] can be applied. The left hand side of the inequality in Equation (2.8) can be approximated by a Gaussian distribution with zero mean, variance NM. Therefore, we can find the probability of the event in Equation (2.8) as follows:

$$\begin{aligned}
p &\equiv \Pr\left\{\left(\mathcal{T}\,\mathbf{u}^{(l)}\right)_i \neq v_i^{(l)}\right\} \\
&= \frac{1}{\sqrt{2\pi NM}}\int_N^{\infty} e^{\frac{t^2}{2NM}}\,dt \\
&\simeq Q\left(\sqrt{\frac{N}{M}}\right),
\end{aligned} \tag{2.9}$$

where

$$Q(x) \equiv \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{\frac{t^2}{2}}\, dt.$$

Next we can find the misclassification rate of the new classifier (P_e) by summing the probabilities of all cases that have no less than $\lfloor (L-\epsilon)/4 \rfloor$ errors. Assuming that $\epsilon \ll L$ yields

$$P_e = \sum_{i=\lfloor \frac{L}{4} \rfloor}^{L} \binom{L}{i} p^i (1-p)^{L-i}. \tag{2.10}$$

Because we are interested in the asymptotic behavior when L is large, the integer floor function can be dropped for clarity. Since in general it is not possible to express Equation (2.10) analytically, we use the Chernoff method to bound P_e from above. Multiplying each term in the sum by a number larger than unity ($e^{t(i-\frac{L}{4})}$, with $t > 0$) and summing from $i = 0$ instead of $i = \frac{L}{4}$, we obtain

$$P_e < \sum_{i=0}^{L} \binom{L}{i} p^i (1-p)^{L-i} e^{t(i-\frac{L}{4})}, \qquad \forall\ t > 0. \tag{2.11}$$

Let $G(t)$ be the right hand side of Equation (2.11), then

$$\begin{aligned} G(t) &= e^{\frac{-Lt}{4}} \sum_{i=0}^{L} \binom{L}{i} \left(pe^t\right)^i (1-p)^{L-i} \\ &= e^{\frac{-Lt}{4}} (1-p+e^t p)^L. \end{aligned} \tag{2.12}$$

Since Equation (2.11) is true for all positive t, we can achieve a tighter bound for the misclassification rate by finding the smallest $G(t)$. So set the derivative of $G(t)$ with respect to t to zero and solve for the optimal t_o,

$$\begin{aligned} G'(t_o) &= 0 \\ \Rightarrow \quad \left(1-p+e^{t_o}p\right)^L &= 4pe^{t_o} \left(1-p+e^{t_o}p\right)^{L-1} \\ \Rightarrow \quad e^{t_o} &= \frac{1-p}{3p}. \end{aligned} \tag{2.13}$$

The condition that $t_o > 0$ implies that $p < 1/4$, which is automatically satisfied since we are dealing with the case when p is small. Substituting the optimal t_o in Equation (2.11) and Equation (2.12) yields

$$\begin{aligned} P_e &< \left(\frac{4}{3}\right)^L \left(\frac{3p}{1-p}\right)^{\frac{L}{4}} (1-p)^L \\ &= \gamma^L \, p^{\frac{L}{4}} \, (1-p)^{\frac{3L}{4}}, \end{aligned} \qquad (2.14)$$

where $\gamma = \dfrac{4}{3^{0.75}} = 1.7547654$.

From Equation (2.9) and (2.14), we can determine the relation between M, the number of classes that can be classified with negligible misclassification rate when the input is one of the prototypes, and the misclassification rate (P_e). Suppose that $P_e = \delta$ and δ is a fixed small number; then

$$\delta^{\frac{4}{L}} < \gamma^4 \, p \, (1-p)^3$$

and

$$\begin{aligned} p &= Q\left(\sqrt{\frac{N}{M}}\right) \\ &> \gamma^{-4} (1-p)^{-3} \, \delta^{\frac{4}{L}}. \end{aligned} \qquad (2.15)$$

For small t, we have $Q^{-1}(t) \simeq \sqrt{2\log(1/t)}$. As N, L, M approach infinity we have

$$\begin{aligned} M &> N \left[Q^{-1}\left(\gamma^{-4} (1-p)^{-3} \, \delta^{\frac{4}{L}}\right)\right]^{-2} \\ &= \frac{N}{8\log\gamma + 6\log(1-p) - (8/L)\log\delta} \\ &\simeq \frac{N}{8\log\gamma} \\ &= 0.22\, N. \end{aligned} \qquad (2.16)$$

From the above lower bound for M, one can easily see that this new classifier is able to store more than a constant times N prototypes and perform minimum Hamming distance classification. Also, since (N/M) stays bounded, the approximation leading to Equation (2.9) is valid. Although the above analysis is done when N, L, and M approach infinity, yet the simulation results in the next section shows that when N, L, and M are moderately large (e. g., 63), the above lower bound applies.

For the case when the pseudoinverse method is used to construct the connection matrix for T, we see no way of analyzing the performance in general, but conjecture that it will be better than the sum-of-outer-product case, especially when the prototypes are correlated.

2.7 Simulation Results and A Character Recognition Example

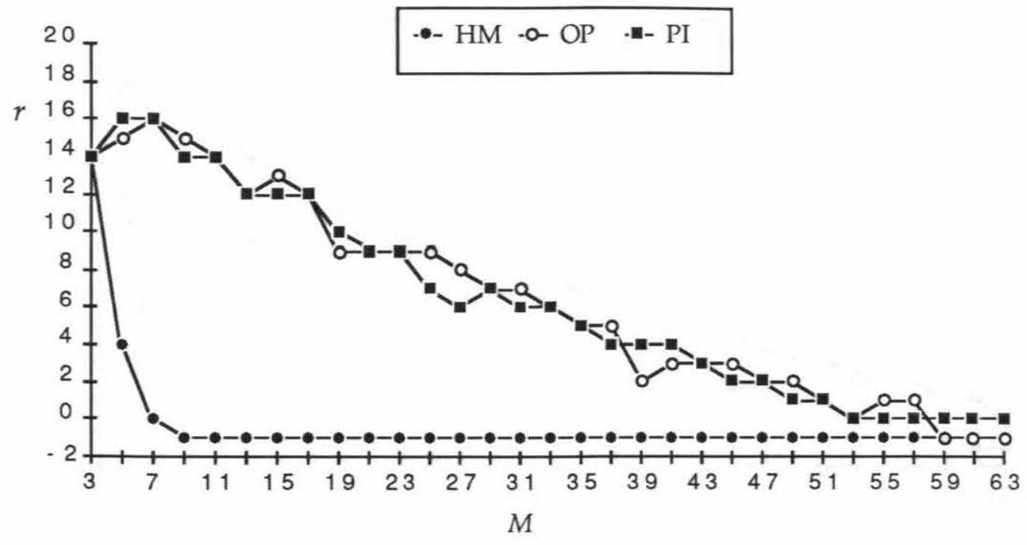
We have simulated both the Hopfield-memory-based classifier and the proposed new classifier (using maximal-length sequence codes) for the cases $L = N = 63$ and $L = N = 127$. Six different classifiers considered are :

- a Hopfield-memory-based classifier
- a new classifier whose connection matrix is generated by the sum-of-outer-product method.
- a new classifier whose connection matrix is generated by the pseudoinverse method.
- a Hopfield-memory-based classifier with clipped connection weights $(-1, 0, 1)$.
- a new classifier whose connection matrix is generated by the sum-of-outer-product method and the components of the connection matrix are clipped $(-1, 0, +1)$.
- a new classifier whose connection matrix is generated by the pseudoinverse method and the components of the connection matrix are clipped $(-1, 0, +1)$.

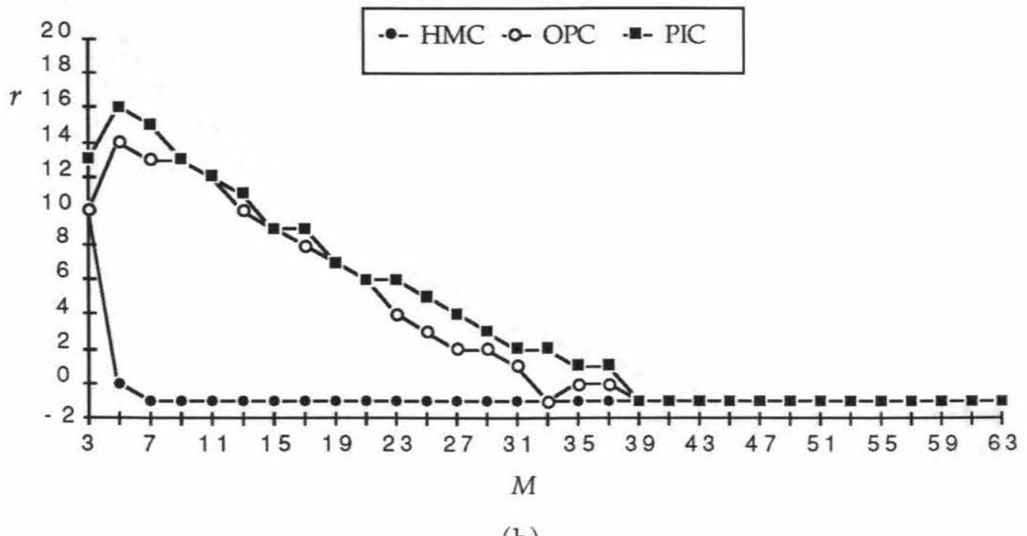
For each case and each choice of N, the program fixes M and the number of errors in input patterns, then randomly generates 10 sets of M prototypes and computes the connection matrix for each classifier. For each classifier it randomly picks a prototype, and adds noise to the prototype by randomly complementing a specified number of bits to get a trial input pattern. In total, 100 trial input patterns are generated for each set of M prototypes. It then feeds the trial inputs to those six classifiers under consideration, checks to see whether the inputs are classified to their respective nearest classes, and reports the percentage of success for each classifier. The simulation results are shown in Figure 2.6 and 2.7. In all figures, the horizontal axis is the number of loaded prototypes (M) and the vertical axis is the attraction radius (r) — the largest number of bit errors allowed. The data are obtained by collecting only those cases where the success rate is more than 99.8%. Here we use the attraction radius of -1 to indicate that for this particular M, with input patterns being prototypes, the success rate is less than 99.8%.

In all cases, our classifier exceeds the performance of the Hopfield-memory-based classifier in terms of storage capacity. For example, consider the case of $N = L = 63$ and clipped connection weights; we find that for an attraction radius of zero, that is, no error in the input vector, the Hopfield-memory-based classifier has a classification capacity of approximately 5, while our new model can store 37. In the case of $N = L = 127$, clipped weights, and zero attraction radius, we have 7 versus 83.

We also notice that the conjectured superiority of the pseudoinverse method over the sum-of-outer-product method is not obvious. The reason for this is that the pseudoinverse method is best for decorrelating the dependency among prototypes, yet the prototypes in this experiment are generated randomly and are presumably independent. Consequently, the advantage of the pseudoinverse method is not obvious from the simulation results. For problems with correlated prototypes, we expect the pseudoinverse method to do much better (see next example).

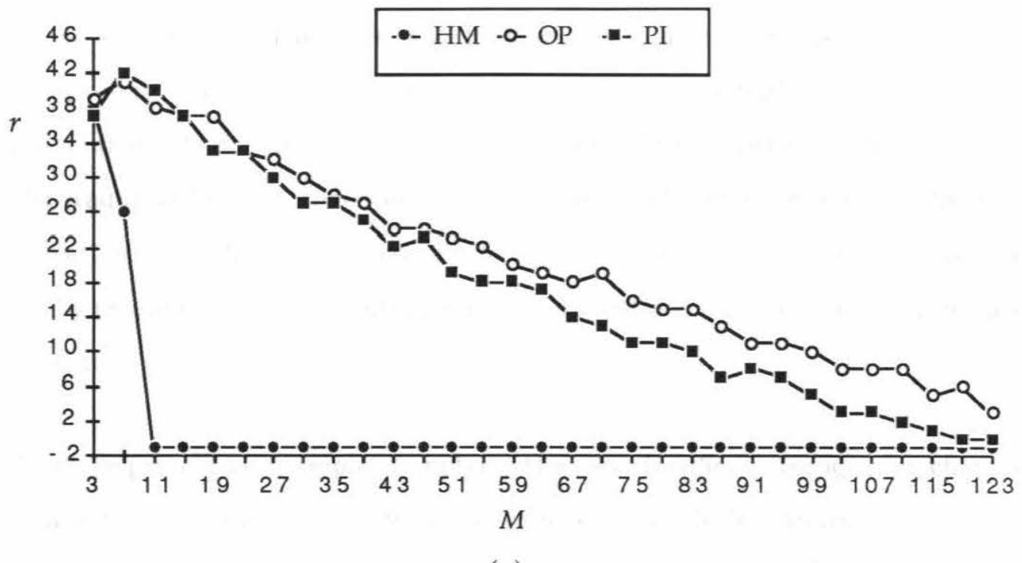


(a)

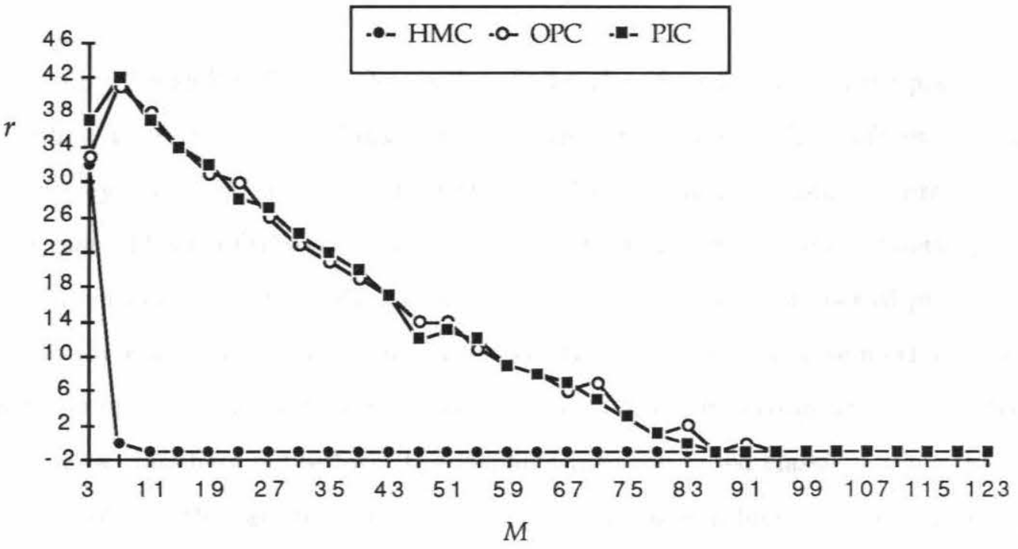


(b)

Figure 2.6: Comparison of performance of the Hopfield-memory-based classifier and the new classifier for $N = 63$ and (a) original connection weights (b) clipped connection weights. HM is the Hopfield-memory-based classifier, OP is the sum-of-outer-product scheme, and PI is the pseudoinverse scheme.



(a)



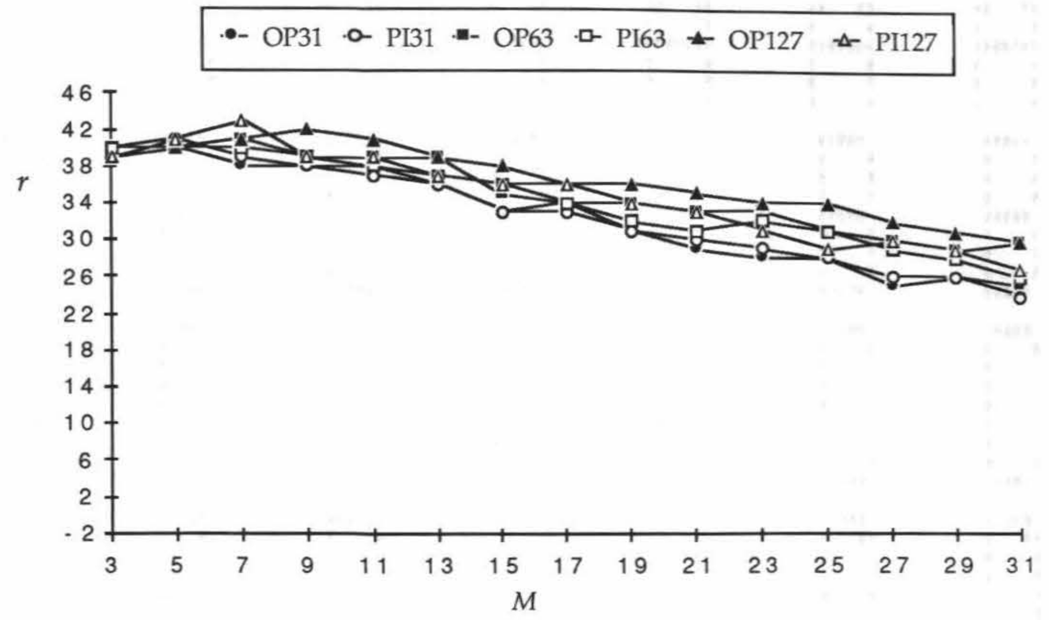
(b)

Figure 2.7: Comparison of performance of the Hopfield-memory-based classifier and the new classifier for $N = 127$ and (a) original connection weights (b) clipped connection weights. HM is the Hopfield-memory-based classifier, OP is the sum-of-outer-product scheme, and PI is the pseudoinverse scheme.

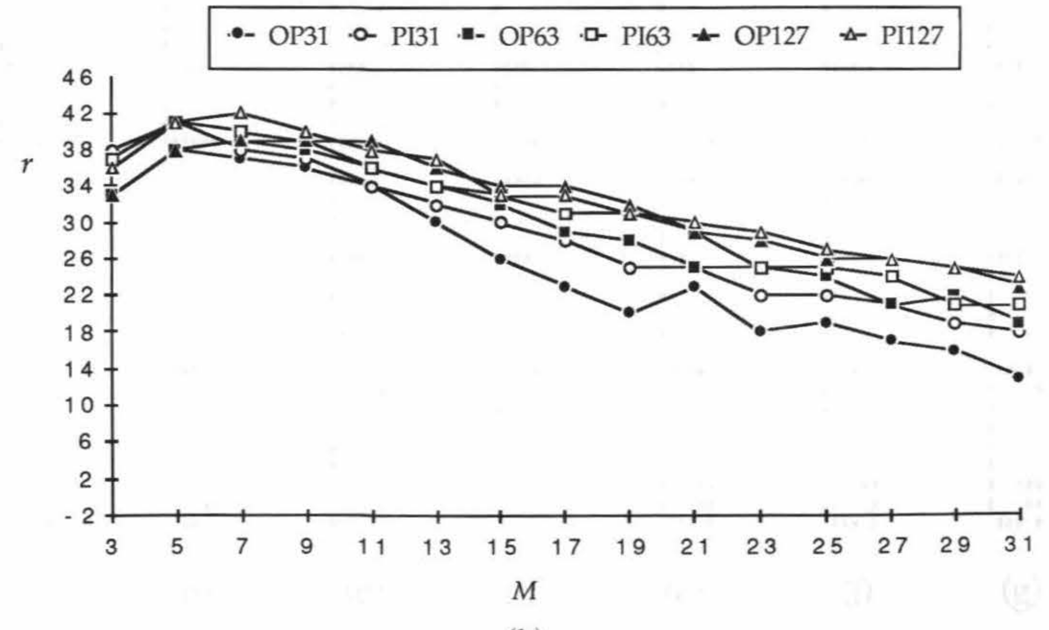
Another simulation using $N = 127$ and successively shorter codes ($L = 127$, $L = 63$, and $L = 31$) reveals that by shortening the code used, the performance of the new classifier degrades only slightly as long as the number of prototypes is less than the code length (see Figure 2.8). Therefore, we think that it is possible to use traditional error correcting codes (e. g., the BCH code) as "internal representation" code. Actually, there have been attempts to use some codes (random or deterministic) with a higher rate than the Hadamard matrix codes and the maximal-length sequence codes as the internal representation code [3, 17, 8]. However, by going to a higher-rate code, one is trading complexity (number of hidden units) for minimum distance, and will possibly get poorer performance.

Next we present an example of applying the new classifier to recognizing characters. Each character is represented by a 9×7 binary pixel array. Noisy inputs are generated by flipping each pixel with 0.1 and 0.2 probability. An input pattern is fed to five classifiers: a Hopfield-memory-based classifier, the new classifiers using the pseudoinverse method and the sum-of-outer-product method with $L = 7$ and $L = 31$.

Figures 2.9 and 2.10 show the results of all 5 classifiers for 0.1 and 0.2 pixel flipping probability, respectively. A blank output means that the classifier refuses to make a decision (rejection). At first, note that the $L = 7$ case is not necessarily worse than the $L = 31$ case. This confirms the earlier conjecture that the performance of networks with fewer hidden units (shorter codes) is only slightly poorer if the number of prototypes is less than the code length. Also, one easily sees that the pseudoinverse method is better than the sum-of-outer-product method because of the correlation among prototypes. All four new classifiers outperform the Hopfield-memory-based classifier since the latter mixes prototypes that are to be remembered and produces a blend of prototypes rather than the prototypes themselves; accordingly, it cannot classify input patterns without mistakes.



(a)



(b)

Figure 2.8: Effects of using codes of different lengths, $N = 127$ and (a) original connection weights (b) clipped connection weights. OP is the sum-of-outer-product scheme and PI is the pseudoinverse scheme. The number is the code length; i. e., $L = 31$, 63, and 127.

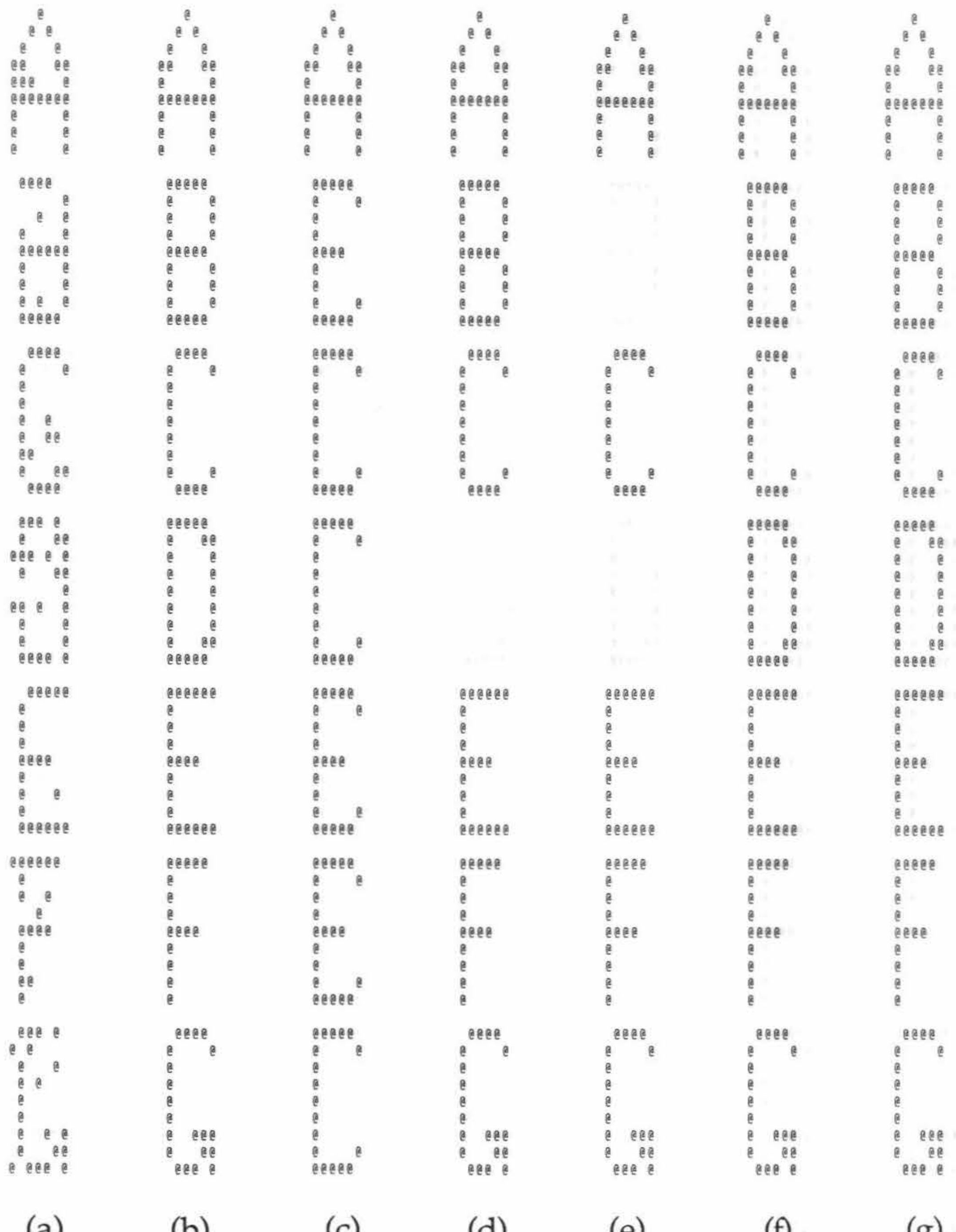


Figure 2.9: The character recognition example with 10% pixel flipping probability in the input. Column (a) is the input, (b) correct output, (c) the Hopfield-memory-based classifier, (d) the new classifier with sum-of-outer-product connection matrix and $L = 7$, (e) same as (d), but $L = 31$, (f) the new classifier with pseudoinverse connection matrix and $L = 7$, (g) same as (f), but $L = 31$.

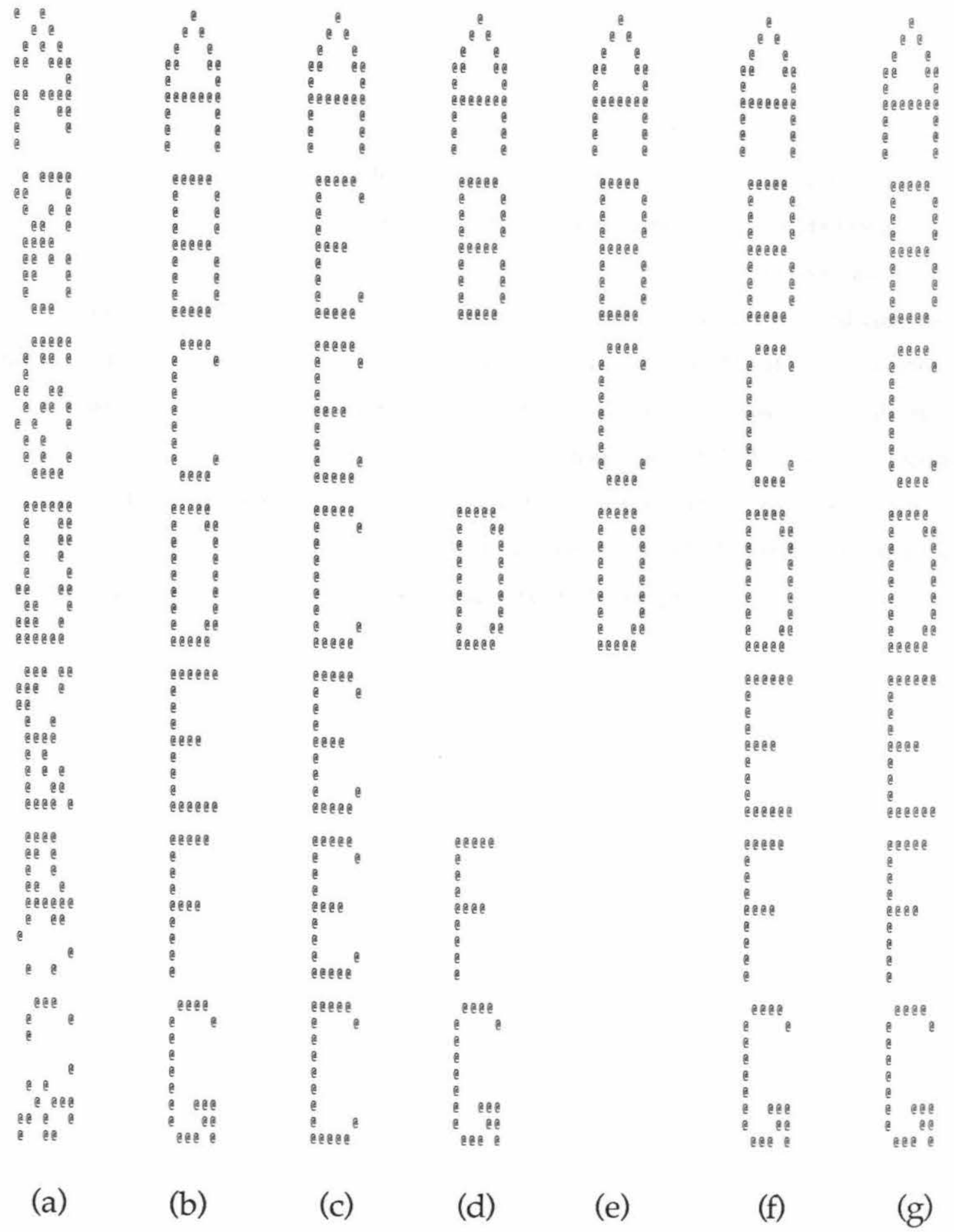


Figure 2.10: The character recognition example with 20% pixel flipping probability in the input. Column (a) is the input, (b) correct output, (c) the Hopfield-memory-based classifier, (d) the new classifier with sum-of-outer-product connection matrix and $L = 7$, (e) same as (d), but $L = 31$, (f) the new classifier with pseudoinverse connection matrix and $L = 7$, (g) same as (f), but $L = 31$.

2.8 Conclusions

In this chapter we have presented a new neural network classifier based on coding theory techniques. The classifier uses codewords from an error correcting code as its "internal representation." Two classes of codes that have good performance are the Hadamard matrix codes and the maximal-length sequence codes. In performance terms we have shown that the new classifier is significantly better than the Hopfield-memory-based classifier. One should also note that when comparing the new classifier with the Hopfield-memory-based classifier, the enhanced performance of the new classifier does not entail extra complexity, since it needs only $L + M$ hard-limiter neurons and $L(N + M)$ connection weights versus N neurons and N^2 weights in a Hopfield memory. In conclusion, we believe that our model forms the basis of a fast, practical method of classification with an efficiency greater than other previous neural network techniques.

References

[1] Y. S. Abu-Mostafa and J. St. Jacques, "Information Capacity of The Hopfield Model," *IEEE Tran. on Information Theory*, Vol. IT-31, 461–464, 1985.

[2] E. R. Berlekamp, *Algebraic Coding Theory*, revised edition. Laguna Hills, CA : Aegean Park Press, 1984.

[3] E. B. Baum, J. Moody, and F. Wilczek, "Internal Representation for Associative Memory," *Biological Cybernetics*, Vol. 59, 217–228, 1988.

[4] W. Feller, *An Introduction to Probability Theory and its Application*, Vol. II, 2nd edition. New York, NY : John Wiley and Sons, 1971.

[5] S. W. Golomb, *Shift Register Sequences.* San Francisco, CA : Holden-Day, 1967.

[6] J. J. Hopfield, "Neural Network and Physical Systems with Emergent Collective Computational Abilities," *Proc. Nat. Acad. Sci. USA*, Vol. 79, 2554–2558, 1982.

[7] J. J. Hopfield, "Neurons with Graded Response Have Collective Computational Properties Like Those of Two-State Neurons," *Proc. Nat. Acad. Sci. USA*, Vol. 81, 3088–3092, 1984.

[8] P. Kanerva, "Parallel Structure in Human and Computer Memory," in *Neural Networks for Computing*, J. S. Denker (editor), New York, NY : American Institute of Physics, 1986, pp. 247–258.

[9] T. Kohonen, *Associative Memory : A System Theoretic Approach.* Berlin : Springer-Verlag, 1977.

[10] T. Kohonen, "Correlation Matrix Memory," in *Neurocomputing — Foundation of Research*,J. A. Anderson and E. Rosenfeld (editor), Cambridge, MA : MIT Press, 1988.

[11] J. P. Lazzaro, S. Ryckebusch, M. A. Mahowald, and C. A. Mead, " Winner-Take-All Networks of $O(N)$ Complexity," in *Advances in Neural Information Processing Systems*, Vol. 1, Palo Alto, CA : Morgan Kaufmann Publishers, 1989, pp. 703-711.

[12] R. Lippmann, "An Introduction to Computing with Neural Nets," *IEEE ASSP Magazine*, Vol. 4, 4–22, 1987.

[13] E. Majani, R. Erlanson, and Y. Abu-Mostafa, " On the K-Winner-Take-All Network," in *Advances in Neural Information Processing Systems*, Vol. 1, Palo Alto, CA : Morgan Kaufmann Publishers, 1989, pp. 634–642.

[14] R. J. McEliece, E. C. Posner, E. R. Rodemich, and S. S. Venkatesh, "The Capacity of The Hopfield Associative Memory," *IEEE Tran. on Information Theory*, Vol. IT-33, 461–482, 1987.

[15] M. L. Minsky and S. A. Papert, *Perceptrons : An Introduction to Computational Geometry*, expanded edition. Cambridge, MA : MIT Press, 1988.

[16] D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing, Explorations in the Microstructure of Cognition*, Vol. I, Cambridge, MA : MIT Press, 1987.

[17] A. Thakoor, "Content-Addressable, High-Density Memories Based on Neural Network Models," Technical Report, Jet Propulsion Lab., JPL D-4166,1987.

Chapter 3

A General Model for Neural Network Associative Memories

3.1 Introduction

Because of their wide usage in many information processing systems, associative memories have attracted much attention from neural network researchers. Unlike traditional computer memory, associative memories have some error correcting capability because they can take in some partially wrong patterns as stimuli and still generate accurate responses. Associative recall is present in many ordinary information processing tasks taking place all the time in human brains, such as recognizing typographic errors, recalling songs' titles, recollecting friends' telephone numbers, etc. As is exemplified by the above examples, the recall process is usually achieved by evolution, through which missing information is gradually filled in until perfect recall is finally reached. It is probably due to the above observation that most proposals for associative memories are based on a feedback architecture.

In this chapter, we will introduce a general model for neural network associative memories. With this model, new associative memories suitable for implementation in other technologies can easily be introduced. An important class of associative memories rely on correlation measures, so they are called correlation associative memories (CAMs). Discussion of CAMs will be deferred until Chapter four. In this chapter, the general model is first presented, and a heuristic justification for its effectiveness is given. The next part of this chapter deals with four representative neural network associative memory models : the Kanerva memory [4, 3], the BMW distributed associative memory [2], the Hamming network associative memory [5], and the spectral associative memory [7]. We will briefly introduce these models and show that each one of them can be formulated as a special case of the proposed general model. Also in each case, the hardware complexity, the capacity, and the stability of the model under consideration will be investigated and reported.

3.2 The General Model

The general model proposed here is intended for autoassociative memories that remember N-bit binary patterns. Suppose $\mathbf{u}^{(1)}, \mathbf{u}^{(2)}, \cdots, \mathbf{u}^{(M)}$ are the M memory patterns to be stored in the associative memory under consideration. Also assume there are a set of M corresponding L-bit codewords, $\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \cdots, \mathbf{v}^{(M)}$ and a matrix $\mathbf{T}$ for mapping the M memory patterns to the corresponding codewords. Note that components of the memory patterns are either $+1$ or -1, while components of the codewords are either 1 or 0. The model is essentially a finite state machine with a state pattern that gets updated at each timestep. To find the next state pattern, the following evolution equation is applied to the current state pattern:

$$\mathbf{x}' = sgn\left\{ \sum_{k=1}^{M} f_k\left(\mathbf{x}, \mathbf{u}^{(1)}, \cdots, \mathbf{u}^{(M)}, \mathbf{v}^{(1)}, \cdots, \mathbf{v}^{(M)}, \mathbf{T}\right) \mathbf{u}^{(k)} \right\}, \tag{3.1}$$

where $\mathbf{x}$ and $\mathbf{x}'$ are the current and the next state patterns, respectively. The recalling process is done by initializing the memory with an input pattern and applying the above evolution equation repetitively until the memory reaches a stable state, which is the final output response.

Now let us give an insightful explanation for the formula in Equation (3.1). Since the objective of an associative memory is to find the nearest memory pattern in some distance measure to the input pattern, it is obvious that the above mechanism will perform correct associative recall if the following three conditions are met : 1) all M memory patterns are stables states, 2) any evolution step brings the next state pattern closer to the correct (nearest) memory pattern, and 3) there is no spurious stable state in the path from the initial input to the nearest memory pattern. Most associative memories do not satisfy the third condition, since there always seems to be numerous spurious stable states in all associative memory systems. Most effort in designing a good associative memory is spent on meeting the first two conditions. As for the third condition, one can only hope that the input is close to the nearest memory pattern so that it is very unlikely that the memory gets stuck at an intermediate spurious stable state.

What Equation (3.1) does is actually very similar to holding an election for each component of the state pattern in order to determine the polarity of that component at the next timestep. For clarity, let us consider only the first component of the next state pattern, x'_1. Each memory pattern (group) has a polarity of its first component, $u_1^{(k)}$ (the preferred party of a group). The strength of the k^{th} memory (the number of votes in a group) pattern is decided by a weighting function f_k. To determine the polarity of x'_1 (the winning party of a citywide election), one takes a weighted sum of all $u_1^{(k)}$'s (counts the votes).

The weighting functions, f_k's, depend on the current state pattern $\mathbf{x}$, all M memory patterns, all M codewords, and the matrix $\mathbf{T}$. If f_k is designed in such a way that its value is large when $\mathbf{x}$ is close to $\mathbf{u}^{(k)}$ and small when they are far apart, then one sees that after one iteration (election), the state pattern will tend to be nearer the nearest memory pattern than it used to be. This is because the nearest memory pattern can be so strong that it overpowers all the others in most of the N elections, so the next state pattern might be almost the same as the nearest memory pattern. To satisfy condition one, one can make f_k extremely large when $\mathbf{x}$ is exactly the same as $\mathbf{u}^{(k)}$; then $\mathbf{u}^{(k)}$ will very likely be a stable state. Therefore, if one carefully designs f_k's and if the memory is not overloaded (more than allowable memory patterns are stored), the associative memory can satisfy the first two conditions.

In the next few sections, we will discuss four representative models and show that they can all be expressed in the form of Equation (3.1). Furthermore, f_k's in those cases all have the property that their value is large when $\mathbf{x}$ and $\mathbf{u}^{(k)}$ are similar and small otherwise. The matrix $\mathbf{T}$ and the M codewords $\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \cdots, \mathbf{v}^{(M)}$ appear in the arguments of f_k only because some models utilize certain coding techniques, otherwise f_k will depend only on $\mathbf{x}$ and $\mathbf{u}^{(1)}, \mathbf{u}^{(2)}, \cdots, \mathbf{u}^{(M)}$. As a matter of fact, for correlation associative memories based on the ordinary correlation, f_k relies only on $\mathbf{x}$ and $\mathbf{u}^{(k)}$, to be more specific, the correlation of these two patterns, i. e., $f_k = f_k(<\mathbf{x}\,, \mathbf{u}^{(k)}>)$.

3.3 Kanerva Memory

Kanerva proposed an associative memory model that is essentially a parallel processing system consisting of address decoders, counters, binary adders, and threshold circuits [4]. He claimed that his model is similar to the human cerebellar model of Marr's [6] and Albus' [1]. Chou [3] later formulated the Kanerva memory in terms of a two-layer feed-forward neural network as shown in Figure 3.1. There are N input nodes, L hidden neurons, and N output neurons. Further, if one uses the Kanerva memory as an autoassociative memory, then a connection between the input end and the output end can be made. All hidden neurons and all output neurons are hard-limiter neurons. The only difference is that the hidden neurons have the following response

$$step(t) \equiv \frac{1}{2}\left(sgn(t-s)+1\right) \equiv \begin{cases} 1 & t \geq s \\ 0 & t < s \end{cases}, \tag{3.2}$$

while the response of the output neurons is the *sgn* function.

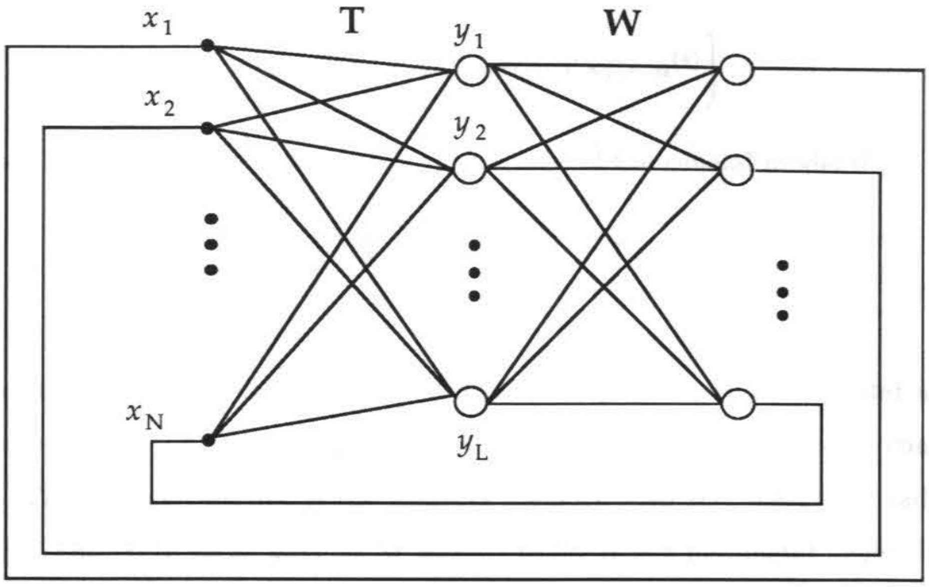


Figure 3.1: Configuration of the Kanerva memory

The first connection matrix $\mathbf{T}$ is an $L \times N$ matrix randomly populated with $+1$ and -1 with equal probability. Assume that $\mathbf{u}^{(k)}, k = 1, 2, \ldots, M$ are the M N-bit memory patterns with $+1$ or -1 components; then the corresponding M L-bit codewords $\mathbf{v}^{(k)}, k = 1, 2, \ldots, M$ are defined as

$$\mathbf{v}^{(k)} \equiv step\left(\mathbf{T}\,\mathbf{u}^{(k)}\right), \qquad k = 1, 2, \ldots, M. \tag{3.3}$$

The second connection matrix $\mathbf{W}$ is constructed by the sum of the outer products of the M memory patterns and the M codewords; i. e.,

$$\mathbf{W} = \sum_{k=1}^{M} \mathbf{u}^{(k)}\,\mathbf{v}^{(k)^t}. \tag{3.4}$$

The evolution equation of the Kanerva memory as an autoassociative memory is

$$\mathbf{x}' = sgn\left\{\mathbf{W} \cdot step\left(\mathbf{T}\,\mathbf{x}\right)\right\}, \tag{3.5}$$

where $\mathbf{x}$ and $\mathbf{x}'$ are the current and the next state patterns.

Substituting Equation (3.4) in (3.5) yields

$$\mathbf{x}' = sgn\left\{\sum_{k=1}^{M} <\mathbf{v}^{(k)}, step(\mathbf{T}\mathbf{x})> \mathbf{u}^{(k)}\right\}. \tag{3.6}$$

Consequently, the Kanerva memory is a special case of the general model if

$$f_k(\mathbf{x}, \mathbf{v}^{(k)}, \mathbf{T}) = <\mathbf{v}^{(k)}, step(\mathbf{T}\mathbf{x})> .$$

The hardware complexity of the Kanerva memory is $L + N$ hard-limiter neurons, LN binary ($+1$ or -1) connection weights, and LN discrete ($-M$ to M) connection weights. The storage capacity of the Kanerva memory with zero attraction radius was shown by Chou [3] to be as large as $M = 2^{\alpha N}$, where α is a parameter depending on s and L. It is also shown that M can be no more than L — the number of hidden neurons. This implies that in order to get exponential capacity, exponential hardware complexity is necessary. To be more specific,

$$LN + LN \log 2M \geq 2^{cN}(cN^2 + 2N)$$

bits are needed to store 2^{cN} memory patterns. There has been virtually no treatment of the stability issue of the Kanerva memory. In Appendix 3.A, we prove that with some assumption, the Kanerva memory is asymptotically stable in both synchronous and asynchronous updating modes.

3.4 BMW Associative Memory

The BMW associative memory model proposed by Baum, Moody, and Wilczek [2] is similar to the Kanerva memory, in that they both use a hidden layer to encode the input pattern before computing the next state pattern. Therefore, the BMW memory has the same architecture as the Kanerva memory (see Figure 3.1). Nevertheless, they differ in how the codes ("internal representations") are specified. In the Kanerva memory, the codewords are obtained by feedinging the M memory patterns to the first layer, while in the BMW memory, the codewords are specified explicitly. In this section we will describe two codes that were discussed in Baum, *et al.*'s work. They are the localized (grandmother cell) coding scheme and the distributed coding scheme.

Again, assume that $\mathbf{u}^{(k)}, \mathbf{v}^{(k)}, k = 1, 2, \ldots, M$ are the M memory patterns and the M corresponding codewords. In the case of the localized coding scheme, $\mathbf{v}^{(k)}$'s are M bits wide ($L = M$) and

$$\mathbf{v}^{(k)} \equiv (0\,0\,\cdots\,1\,\cdots\,0\,0),$$

i. e., the k^{th} unit vector. For the distributed coding scheme, the L bits in a codeword are partitioned into S groups and the l^{th} group has r_l bits, where r_l's are assumed to be relative prime. In the k^{th} codeword, the j^{th} bit of the l^{th} group is

$$v^{(k)}_{t_l+j} \equiv \begin{cases} 1 & k \equiv j \bmod r_l \\ 0 & \text{otherwise} \end{cases},$$

where $t_l = \sum_{i=1}^{l-1} r_i$.

In both cases the matrix $\mathbf{T}$ is constructed by the sum-of-outer-product rule and

$$\mathbf{T} \equiv \sum_{k=1}^{M} \mathbf{v}^{(k)}\, \mathbf{u}^{(k)^t}. \tag{3.7}$$

As in the Kanerva memory, the second connection matrix $\mathbf{W}$ is

$$\mathbf{W} \equiv \sum_{k=1}^{M} \mathbf{u}^{(k)}\, \mathbf{v}^{(k)^t} = \mathbf{T}^t. \tag{3.8}$$

The evolution equation of the BMW memory is

$$\begin{aligned} \mathbf{x}' &= sgn\,\{\,\mathbf{W}\cdot step(\mathbf{T}\,\mathbf{x})\,\} \\ &= sgn\left\{ \sum_{k=1}^{M} <\mathbf{v}^{(k)},\, step(\sum_{l=1}^{M} <\mathbf{u}^{(l)}, \mathbf{x}>\, \mathbf{v}^{(l)}\,)>\, \mathbf{u}^{(k)} \right\}. \end{aligned} \tag{3.9}$$

From the above equation, it is clear that if we let

$$f_k(\mathbf{x}, \mathbf{u}^{(1)}, \cdots, \mathbf{u}^{(M)}, \mathbf{v}^{(1)}, \cdots, \mathbf{v}^{(M)}) \equiv <\mathbf{v}^{(k)},\ step(\sum_{l=1}^{M} <\mathbf{u}^{(l)}, \mathbf{x}> \mathbf{v}^{(l)})>,$$

then the BMW memory is a special case of the general associative memory model.

The hardware complexity required to implement the BMW memory is $L+N$ hard-limiter neurons and LN discrete connection weights ($-M$ to $+M$, here only one set of connection weights is needed since $\mathbf{W}$ is the transpose of $\mathbf{T}$). The capacity of the BMW memory has been shown to be less than L [2]. The BMW model was originally proposed as a memory based on a feed-forward network with one hidden layer; hence very little work has been done on the stability of the feedback BMW memory. Since the only difference between the BMW memory and the Kanerva memory is the way that the codes are prescribed, we expect them to behave similarly in terms of stability. We show, in Appendix 3.A, that with some assumption the BMW memory is asymptotically stable.

3.5 Hamming Network Associative Memory

The Hamming network [5] is similar to the BMW model with the unary (grandmother cell) code, except that the hidden neurons are of the sigmoidal type and there are mutually inhibitory connections among hidden neurons. The hidden layer functions as a winner-take-all circuit, and only the hidden neuron with maximum input will be activated (+1), while all the other hidden neurons will be off (0). The two connection matrices of the Hamming network associative memory $\mathbf{T}$ and $\mathbf{W}$ are exactly the same as those of the BMW memory with the unary code.

The evolution equation of the Hamming network associative memory is

$$\mathbf{x}' = sgn\left\{\sum_{k=1}^{M} \delta_{kl}\, \mathbf{u}^{(k)}\right\}, \tag{3.10}$$

where

$$<\mathbf{u}^{(l)}, \mathbf{x}> = \max_j <\mathbf{u}^{(j)}, \mathbf{x}>$$

and δ_{kl} is the Kronecker delta function. Therefore, if we define

$$f_k(\mathbf{x}, \mathbf{u}^{(1)}, \cdots, \mathbf{u}^{(M)}) \equiv \begin{cases} 1 & <\mathbf{u}^{(k)}, \mathbf{x}> = \max_j <\mathbf{u}^{(j)}, \mathbf{x}> \\ 0 & \text{otherwise} \end{cases},$$

then the Hamming network associative memory becomes an instance of the general model.

The hardware complexity needed to implement the Hamming network associative memory is M sigmoidal neurons, N hard-limiter neurons, and $M^2 + MN$ connection weights. The storage capacity of the Hamming network is actually limited only by the efficiency of the winner-take-all circuit. As the number of hidden neurons grows, the mutually inhibitory feedback network gradually deteriorates in response time and accuracy. As for the stability, since the mutually inhibitory network is stable, we conclude that the Hamming network associative memory is asymptotically stable.

3.6 Spectral Associative Memory and Pseudoinverse Associative Memory

The spectral associative memory [7] is based on a fully-connected network architecture with N neurons and the following $N \times N$ connection weight matrix,

$$\mathbf{T} \equiv \mathbf{U}\,\mathbf{\Lambda}\left(\mathbf{U}^t\,\mathbf{U}\right)^{-1}\mathbf{U}^t, \tag{3.11}$$

where $\mathbf{U} \equiv (\mathbf{u}^{(1)}, \mathbf{u}^{(2)}, \cdots, \mathbf{u}^{(M)})$ and $\mathbf{\Lambda}$ is an $M \times M$ diagonal matrix whose entries are $\lambda_1, \lambda_2, \cdots, \lambda_M$. Substituting Equation (3.11) in the evolution equation of a fully-connected network gives

$$\begin{aligned}
\mathbf{x}' &= sgn\left\{\mathbf{T}\,\mathbf{x}\right\} \\
&= sgn\left\{\mathbf{U}\,\mathbf{\Lambda}\,\left(\mathbf{U}^t\,\mathbf{U}\right)^{-1}\,\mathbf{U}^t\,\mathbf{x}\right\} \\
&= sgn\left\{\mathbf{U}\,\mathbf{\Lambda}\,\left(\mathbf{U}^t\,\mathbf{U}\right)^{-1}\,\left(<\mathbf{u}^{(1)}\,,\,\mathbf{x}> \, <\mathbf{u}^{(2)}\,,\,\mathbf{x}> \,\cdots\, <\mathbf{u}^{(M)}\,,\,\mathbf{x}>\right)^t\right\} \\
&= sgn\left\{\mathbf{U}\,\mathbf{\Lambda}\,\left(\mathbf{U}^t\,\mathbf{U}\right)^{-1}\,\mathbf{w}\right\}, \qquad (3.12)
\end{aligned}$$

where $\mathbf{w}$ is an $M \times 1$ vector and $w_k = <\mathbf{u}^{(k)}\,,\,\mathbf{x}>$. Let $\mathbf{V} \equiv (\mathbf{U}^t\,\mathbf{U})^{-1}$ and $\mathbf{v}^{(k)}$ be the kth row of $\mathbf{V}$, then

$$\begin{aligned}
\mathbf{x}' &= sgn\left\{\sum_{k=1}^{M} \lambda_k \, <\mathbf{v}^{(k)}, \mathbf{w}> \, \mathbf{u}^{(k)}\right\} \\
&= sgn\left\{\sum_{k=1}^{M} \left[\lambda_k \sum_{j=1}^{M} v_j^{(k)} \left(\sum_{i=1}^{N} u_i^{(k)}\,x_i\right)\right]\,\mathbf{u}^{(k)}\right\} \\
&= sgn\left\{\sum_{k=1}^{M} \left(\lambda_k \sum_{i=1}^{N}\sum_{j=1}^{M} v_j^{(k)}\,u_i^{(j)}\,x_i\right)\,\mathbf{u}^{(k)}\right\}. \qquad (3.13)
\end{aligned}$$

Therefore, if we define

$$f_k\,(\mathbf{x}, \mathbf{u}^{(1)}, \cdots, \mathbf{u}^{(M)}, \mathbf{v}^{(k)}) \equiv \lambda_k \sum_{i=1}^{N}\sum_{j=1}^{M} v_j^{(k)}\,u_i^{(j)}\,x_i,$$

the spectral associative memory model reduces to an instance of the general model proposed in Section 3.2.

Since the spectral associative memory is based on an N-neuron fully-connected network structure, it needs N hard-limiter neurons and N^2 continuous connection weights. As for the storage capacity of the spectral associative memory, Venkatesh showed empirically that it is better than the Hopfield memory. He also proved that the storage capacity of the spectral associative memory is N when N is large [7]. AS far as the stability is concerned, Venkatesh also showed that by using

$$E(\mathbf{x}) \equiv <\mathbf{x}\,,\,\mathbf{T}\,\mathbf{x}>$$

as the energy function, the spectral associative memory will converge to a stable state in both synchronous and asynchronous updating modes.

As for the pseudoinverse associative memory, since it is a special case of the spectral associative memory with the matrix $\mathbf{\Lambda}$ equal to the identity matrix, all the above results apply to it.

3.7 Correlation Associative Memories

There are many other associative memory models based on correlation (similarity measures), such as the correlation-matrix associative memory, the high-order correlation associative memory, the potential function correlation associative memory, the exponential associative memory, and the associative memories based on the generalized correlation measure. We will defer the discussion of these models until the next chapter.

3.8 Conclusions

In this chapter we propose a general model that subsumes most neural associative memory models with feedback architecture reported in Chapter one. It is based on an algorithm similar to the political election process. In the evolution equation, a weighted sum of all memory patterns is first calculated; then a decision is made to determine the polarity of each component of the next state pattern. We select four models — the Kanerva memory, the BMW memory, the Hamming network associative memory, and the spectral associative memory as examples and show that with proper choice of weighting functions, they are all special cases of the general model. Moreover, we also prove that in both asynchronous and synchronous updating modes, the Kanerva memory and the BMW model are both asymptotically stable.

Appendix 3.A Asymptotic Stability of Kanerva Memory and BMW Memory

The strategy used to prove the asymptotic stability of the Kanerva memory is similar to that of Hopfield model. An energy (Liapunov) function is defined and it is shown that after each timestep the energy will not increase. This fact, together with the finiteness of the energy function and the fact that the energy cannot stay at the same level forever, proves that the Kanerva memory is asymptotically stable. At first, let us make the following assumption:

Assumption : Let $\mathbf{x}$ be a binary state pattern and $\mathbf{u}^{(k)}, k = 1, 2, \cdots, M$ be the memory patterns; then $< step(\mathbf{T}\,\mathbf{u}^{(k)}),\, step(\mathbf{T}\,\mathbf{x}) >$ depends only on the distance between $\mathbf{u}^{(k)}$ and $\mathbf{x}$, or equivalently on $< \mathbf{u}^{(k)}, \mathbf{x} >$.

Note that the original interpretation of $<step(\mathbf{T}\,\mathbf{u}^{(k)}),\, step(\mathbf{T}\,\mathbf{x})>$ given by Kanerva is the number of rows in $\mathbf{T}$ that are less than $(N - s + 1)/2$ bits away from both $\mathbf{u}^{(k)}$ and $\mathbf{x}$. Thus, it is apparent that the assumption holds most of the time if N is large and L is not much smaller than 2^N. It is also clear that as $\mathbf{x}$ moves away from $\mathbf{u}^{(k)}$, $< step(\mathbf{T}\,\mathbf{u}^{(k)}),\, step(\mathbf{T}\,\mathbf{x}) >$ will become smaller and smaller.

Now define a sequence of patterns $\mathbf{x}_0^{(k)}, \mathbf{x}_1^{(k)}, \cdots, \mathbf{x}_N^{(k)}$, which constitute a path from $\mathbf{u}^{(k)}$ to $-\mathbf{u}^{(k)}$. Also,

$$\mathbf{x}_0^{(k)} = \mathbf{u}^{(k)}, \quad \mathbf{x}_N^{(k)} = -\mathbf{u}^{(k)},$$

and

$$d_{\text{Hamming}}(\mathbf{u}^{(k)}, \mathbf{x}_i^{(k)}) = i \qquad\qquad i = 0, 1, \ldots, N.$$

Let $d_k = d_{\text{Hamming}}(\mathbf{u}^{(k)}, \mathbf{x})$; and then define the energy associated with $\mathbf{u}^{(k)}$ as

$$E^{(k)}(\mathbf{x}) \equiv \sum_{i=0}^{d_k} < step(\mathbf{T}\,\mathbf{u}^{(k)}),\, step(\mathbf{T}\,\mathbf{x}_i^{(k)}) > . \tag{3.14}$$

Define the total energy of the system at state $\mathbf{x}$ as

$$E(\mathbf{x}) \equiv \sum_{k=1}^{M} E^{(k)}(\mathbf{x}). \tag{3.15}$$

Suppose all neurons in the output layer of the Kanerva memory update themselves according to Equation (3.5); then the difference between the energies of the current and the next states is

$$\begin{aligned} \Delta E &= E(\mathbf{x}') - E(\mathbf{x}) \\ &= \sum_{k=1}^{M} E^{(k)}(\mathbf{x}') - E^{(k)}(\mathbf{x}). \end{aligned} \tag{3.16}$$

Considering only the k^{th} term of the sum in Equation (3.16) and assuming that

$$d_{\text{Hamming}}(\mathbf{u}^{(k)}, \mathbf{x}) = d_k \text{ and } d_{\text{Hamming}}(\mathbf{u}^{(k)}, \mathbf{x}') = d'_k,$$

then

(a) if $d_k \geq d'_k$:

$$\begin{aligned} E^{(k)}(\mathbf{x}') - E^{(k)}(\mathbf{x}) &= -\sum_{i=d'_k}^{d_k} < step(\mathbf{T}\,\mathbf{u}^{(k)}) \, , \, step(\mathbf{T}\,\mathbf{x}_i^{(k)}) > \\ &\leq - < step(\mathbf{T}\,\mathbf{u}^{(k)}) \, , \, step(\mathbf{T}\,\mathbf{x}_{d_k}^{(k)}) > \cdot \, (d_k - d'_k) \\ &\leq - < step(\mathbf{T}\,\mathbf{u}^{(k)}) \, , \, step(\mathbf{T}\,\mathbf{x}) > \cdot \, (d_k - d'_k). \end{aligned}$$

(b) if $d_k < d'_k$:

$$\begin{aligned} E^{(k)}(\mathbf{x}') - E^{(k)}(\mathbf{x}) &= \sum_{i=d_k}^{d'_k} < step(\mathbf{T}\,\mathbf{u}^{(k)}) \, , \, step(\mathbf{T}\,\mathbf{x}_i^{(k)}) > \\ &\leq < step(\mathbf{T}\,\mathbf{u}^{(k)}) \, , \, step(\mathbf{T}\,\mathbf{x}_{d_k}^{(k)}) > \cdot \, (d'_k - d_k) \\ &\leq < step(\mathbf{T}\,\mathbf{u}^{(k)}) \, , \, step(\mathbf{T}\,\mathbf{x}) > \cdot \, (d'_k - d_k). \end{aligned}$$

In both cases, the same inequality holds. Substituting the above inequality in Equation (3.16) yields

$$\begin{aligned}
\Delta E &\leq \sum_{k=1}^{M} <step(\mathbf{T}\,\mathbf{u}^{(k)})\,,\,step(\mathbf{T}\,\mathbf{x})> \cdot (d'_k - d_k) \\
&= \frac{1}{2}\sum_{k=1}^{M} <step(\mathbf{T}\,\mathbf{u}^{(k)})\,,\,step(\mathbf{T}\,\mathbf{x})> \cdot \Big(<\mathbf{u}^{(k)}\,,\,\mathbf{x}> - <\mathbf{u}^{(k)}\,,\,\mathbf{x}'>\Big) \\
&= \frac{1}{2}\left\{\sum_{k=1}^{M} <step(\mathbf{T}\,\mathbf{u}^{(k)})\,,\,step(\mathbf{T}\,\mathbf{x})> \cdot \left(\sum_{j=1}^{N} u_j^{(k)}\,(x_j - x'_j)\right)\right\} \\
&= \frac{1}{2}\sum_{j=1}^{N}\left\{\sum_{k=1}^{M} <step(\mathbf{T}\,\mathbf{u}^{(k)})\,,\,step(\mathbf{T}\,\mathbf{x})> \cdot u_j^{(k)}\right\}(x_j - x'_j) \\
&\leq 0.
\end{aligned} \tag{3.17}$$

We therefore conclude that the Kanerva memory is asymptotically stable in the synchronous updating mode. Similarly, the Kanerva memory is also asymptotically stable in the asynchronous updating mode.

∎

Since the BMW memory is different from the Kanerva memory only in the codes used as internal representations, we need only make some modification of the previous proof to show the asymptotic stability of the BMW memory. The assumption to be made is the same as the previous one except that the measure is changed.

Assumption : The measure $<\mathbf{v}^{(k)}\,,\,step(\sum_{l=1}^{M}\ <\mathbf{u}^{(l)}\,,\,\mathbf{x}>\ \cdot \mathbf{v}^{(l)}\,)>$ depends only on the distance between $\mathbf{u}^{(k)}$ and $\mathbf{x}$, or equivalently, on $<\mathbf{u}^{(k)}\,,\,\mathbf{x}>$.

It is also apparent that as the distance between $\mathbf{x}$ and $\mathbf{u}^{(k)}$ becomes larger, the quantity $<\mathbf{v}^{(k)}\,,\,step(\sum_{l=1}^{M}\ <\mathbf{u}^{(l)}\,,\,\mathbf{x}>\ \cdot \mathbf{v}^{(l)}\,)>$ will become smaller.

Again, we can define a sequence of patterns from $\mathbf{u}^{(k)}$ to $-\mathbf{u}^{(k)}$ and the energy associated with $\mathbf{u}^{(k)}$,

$$E^{(k)}(\mathbf{x}) \equiv \sum_{i=0}^{d_k} < \mathbf{v}^{(k)},\ step(\sum_{l=1}^{M} < \mathbf{u}^{(l)}, \mathbf{x}_i^{(k)} > \cdot \mathbf{v}^{(l)}) > . \tag{3.18}$$

Also define the total energy of the system in state $\mathbf{x}$ as

$$E(\mathbf{x}) \equiv \sum_{k=1}^{M} E^{(k)}(\mathbf{x}). \tag{3.19}$$

Following the same derivation, it can be shown that the energy will stay the same or decrease after every iteration; thus, we conclude that the BMW associative memory model is asymptotically stable in synchronous updating mode. By the same token, the BMW associative memory is also asymptotically stable in the asynchronous updating mode.

$\blacksquare$

References

[1] J. S. Albus, *Brain, Behavior, and Robotics.* Peterborough, NH : BYTE book of McGraw-Hill, 1981.

[2] E. B. Baum, J. Moody, and F. Wilczek, "Internal Representation for Associative Memory," *Biological Cybernetics*, Vol. 59, 217–228, 1988.

[3] P. A. Chou, "The Capacity of the Kanerva Associative Memory is Exponential," in *Neural Information Processing Systems*, D. Z. Anderson (editor), New York, NY : American Institute of Physics, 1988, pp. 184–191.

[4] P. Kanerva, "Parallel Structure in Human and Computer Memory," in *Neural Networks for Computing*, J. S. Denker (editor), New York, NY : American Institute of Physics, 1986, pp. 247–258.

[5] R. Lippmann, "An Introduction to Computing with Neural Nets," *IEEE ASSP Magazine*, Vol. 4, No. 2, 4–22, 1987.

[6] D. Marr, "A Theory of Cerebellar Cortex," *Journal of Physiology*, Vol. 202, 437–470, 1969.

[7] S. S. Venkatesh, "Linear Map with Point Rules," Ph. D. dissertation, California Institute of Technology, 1987.

Chapter 4

Correlation Associative Memories

4.1 Introduction

Since the seminal work of Hopfield [9, 10], there has been much interest in building associative memory systems using neural network approaches. The storage capacity of the Hopfield memory has been found, both empirically [9] and theoretically [12], to scale less than linearly (approximately $N/\log N$) with the number of components in the memory patterns. Lee *et al.* [11], Soffer [15], Psaltis and Park [13], and Dembo and Zeitouni [4, 5] all proposed new architectures that utilize correlation matrices and nonlinear circuits. Previously, we also proposed a new associative memory model that adopts the exponentiation function [2]. All these models can be implemented by a feedback network with two layers: The first layer computes the correlations of the input pattern and all memory patterns and then applies some nonlinear function; the second calculates a weighted sum and thresholds that sum.

In this chapter, we discuss a class of neural network associative memories that are based on correlation measures, which we call correlation associative memories (CAMs). In Section 4.2, we introduce a model for the correlation associative memories, show that this model is subsumed by the general model of feedback neural network associative memories in Chapter three, and demonstrate that some known associative memory models can be formulated as correlation associative memories. Section 4.3 deals with the convergence property of correlation associative memories. First, an energy function is defined, and it is then shown that this energy function will decrease or stay put after every iteration. Thus, we conclude that correlation associative memories converge to stable states in both asynchronous and synchronous updating modes. In Section 4.4, we concentrate on a particular model called exponential correlation associative memory (ECAM) and investigate thoroughly the relationship between the storage capacity and the attraction radius of ECAM. We find that if all input patterns inside a sphere of some attraction radius around a memory pattern need be attracted to that memory pattern with high probability, then the storage capacity is proportional to c^N, where the constant c decreases as the attraction radius increases. More importantly, we find that under some condition the storage capacity of ECAM actually meets the sphere-packing bound in information theory [3]. Also, it is found that the storage capacity of ECAM will be proportional to the dynamic range if the dynamic range of exponentiation circuits in an ECAM is fixed. However, the hardware complexity of an ECAM capable of storing an exponential number of memory patterns is also exponential. In Section 4.5, we present the results of some simulation experiments of ECAM and show that they confirm the theoretical findings about the storage capacity of ECAM, even though N is not excessively large as assumed when proving the theoretical results.

4.2 A Model for Correlation Associative Memories

First, let us introduce the idea of generalized correlation measure : Let $\mathbf{u}$ and $\mathbf{x}$ be two N-bit binary patterns whose components are either $+1$ or -1, and $\mathbf{H}$ be a $N \times N$ real matrix; then the generalized correlation measure of $\mathbf{u}$ and $\mathbf{x}$ weighted by $\mathbf{H}$ is

$$< \mathbf{u} \, , \, \mathbf{x} >_{\mathbf{H}} \;\equiv\; \sum_{i=1}^{N} \sum_{j=1}^{N} u_i \, H_{ij} \, x_j . \tag{4.1}$$

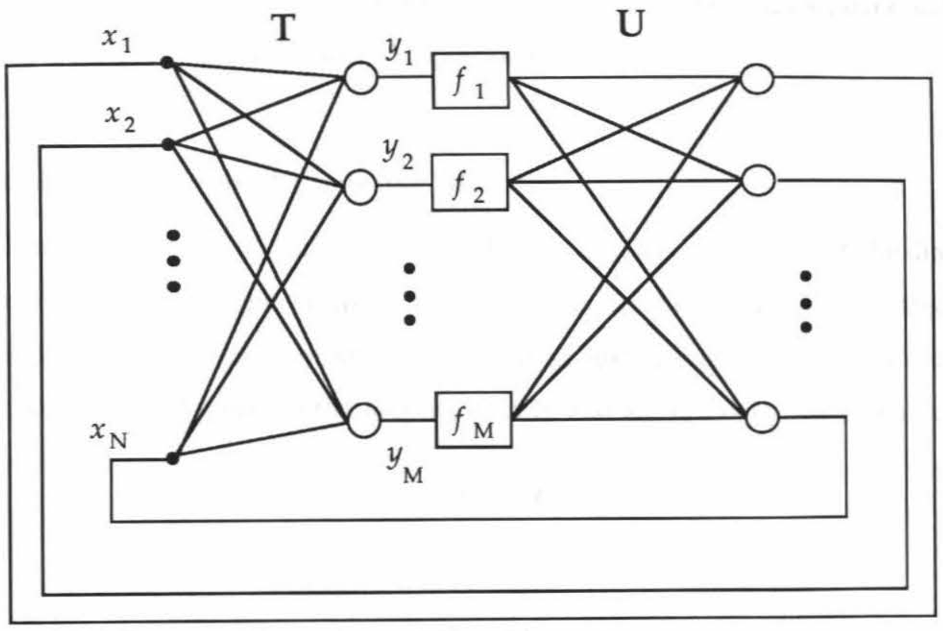


Figure 4.1: Architecture of correlation associative memories

Note that the matrix $\mathbf{H}$ works as a way of adjusting the contributions from N components of $\mathbf{u}$ and $\mathbf{x}$ to the overall correlation measure. Moreover, if $\mathbf{H}$ is the identity matrix, then the generalized correlation measure reduces to the ordinary correlation; that is, $<\mathbf{u}\,,\mathbf{x}>_{\mathbf{I}} \;=\; <\mathbf{u}\,,\mathbf{x}>$. Also note that $<\mathbf{u}\,,\mathbf{x}>= N - 2d_{\text{Hamming}}(\mathbf{u}\,,\mathbf{x})$.

Now, assume that $\mathbf{u}^{(1)}, \mathbf{u}^{(2)}, \cdots, \mathbf{u}^{(M)}$ are the M memory patterns; then the evolution equation of correlation associative memories is

$$\mathbf{x}' \;=\; sgn\left\{\sum_{k=1}^{M} f_k\left(<\mathbf{u}^{(k)}\,,\mathbf{x}>_{\mathbf{H}}\right)\cdot \mathbf{u}^{(k)}\right\}. \tag{4.2}$$

It is apparent from Equation (4.2) that all correlation associative memories are instances of the general associative memory model. Figure 4.1 illustrates a neural network architecture of correlation associative memories. The first connection matrix is

$$\mathbf{T} \;=\; \mathbf{U}^t\mathbf{H},$$

where $\mathbf{U}$ is an $M\times N$ matrix that is made up of M memory patterns $\mathbf{u}^{(k)}, k = 1, 2, \ldots, M$.

Next we will describe how some well known neural network associative memory models can be expressed in the form of Equation (4.2).

- **Correlation-Matrix Associative Memory**

  This model is essentially the same as the Hopfield memory except that the diagonal of the connection weight matrix is not zeroed. It can be easily demonstrated that the correlation-matrix associative memory is an instance of correlation associative memories with $\mathbf{H}$ equal to the identity matrix and all f_k's equal to f, where

$$f(t) \equiv t.$$

- **High-Order Correlation Associative Memory**

  In this type of associative memory, $\mathbf{H}$ is again equal to the identity matrix, but the weighting function is now a nonlinear function,

$$f(t) \equiv t^q, \tag{4.3}$$

  where q is usually an integer and $q > 1$. The capacity of the high-order associative memory is proportional to N^q asymptotically [15], which is usually large enough for all practical purposes.

- **Potential Function Correlation Associative Memory**

  Sayeh and Han [14] and Dembo and Zeitouni[4, 5] independently introduced this model, which is proposed originally for continuous-time systems with real-valued patterns. Still, it is straightforward to express this model in discrete-time formulation. Again, $\mathbf{H}$ is the identity matrix, all f_k's are equal to f, and

$$f(t) = (N - t)^{-L}, \tag{4.4}$$

  where L is an integer and $L > 3$. The capacity of this model has been demonstrated to be limited only by the sphere-packing bound in information theory [5].

The primary disadvantage of this model is that hardware implementation of the nonlinear potential function is very cumbersome.

- **Exponential Correlation Associative Memory (ECAM)**

We introduced a new correlation associative memory that adopts the exponentiation function for all f_k's [2]; i. e.,

$$f(t) \equiv a^t, \tag{4.5}$$

where $a > 1$. Here $\mathbf{H}$ is again equal to the identity matrix. The storage capacity of ECAM will be explored in Section 4.4.

- **Correlation Associative Memory based on the Mahalanobis distance**

The previous four models are all based on the ordinary correlation, namely, correlation measure with $\mathbf{H}$ equal to the identity matrix. Nonetheless, there are times when some distance other than the Hamming distance is more convenient. For example, if an associative memory is used to solve a pattern classification problem with multivariate Gaussian distributions, then the Mahalanobis distance is a better similarity measure than the Hamming distance [6]. The Mahalanobis distance from $\mathbf{u}^{(k)}$ to $\mathbf{x}$ is defined as

$$d_{\text{Mahalanobis}}(\mathbf{u}^{(k)}, \mathbf{x}) \equiv (\mathbf{u}^{(k)} - \mathbf{x})\, \mathbf{\Sigma}^{-1}\, (\mathbf{u}^{(k)} - \mathbf{x}),$$

where we assume that the covariance matrices of all M distributions are the same and equal to $\mathbf{\Sigma}$. After some derivation, it can be shown that

$$< \mathbf{u}^{(k)}, \mathbf{x} >_{\mathbf{\Sigma}^{-1}} = G_1 + G_2^{(k)} - \frac{1}{2}\, d_{\text{Mahalanobis}}(\mathbf{u}^{(k)}, \mathbf{x})$$

where G_1 is a constant depending only on $\mathbf{x}$ and $G_2^{(k)}$ depends only on $\mathbf{u}^{(k)}$. Therefore, a correlation associative memory with $\mathbf{H} = \mathbf{\Sigma}^{-1}$ and appropriate thresholds and weighting functions can be used to solve the minimum Mahalanobis distance classification problem.

4.3 The Convergence Property of CAMs

Since CAMs are based on a feedback network structure, understanding their asymptotic behavior is very crucial, for if a CAM does not settle down to a stable state, it would be impossible to obtain any valid output, rendering it useless.

Hopfield [9] proved that his model is asymptotically stable when running in the asynchronous update mode (only one neuron in the output layer updates itself at a time). At first he introduced an energy (Liapunov) function of the system, and went on to demonstrate that the energy function will only decrease or stay the same after every iteration. Moreover, he showed that the energy function has a lower bound and that the system can not stay at the same energy level forever. These facts together with the nonincreasing property of the energy function imply that the Hopfield memory will eventually reach a stable state with minimum energy level. However, if the Hopfield memory is running in synchronous mode (all neurons in the output layer update themselves at the same time), it may not converge to a fixed point and may become oscillatory between two states [1].

In this section, we prove that correlation associative memories are asymptotically stable in both asynchronous and synchronous updating modes. To begin with, let us introduce a lemma.

Lemma 4.1 : Let $f(t)$ be continuous and monotonically nondecreasing over $[-N, N]$; then a correlation associative memory with the following evolution equation,

$$\mathbf{x}' = sgn\left\{\sum_{k=1}^{M} f(<\mathbf{u}^{(k)}, \mathbf{x}>) \cdot \mathbf{u}^{(k)}\right\}$$

is stable in both synchronous and asynchronous updating modes.

Proof : see Appendix 4.A.

Theorem 4.2 : All four models based on the ordinary correlation in the previous section are stable in both synchronous and asynchronous updating modes.

Proof : Since for all $t_1 > t_2$, we have

$$\begin{aligned} t_1 &\geq t_2, \\ t_1^q &\geq t_2^q, \\ (N-t_1)^{-L} &\geq (N-t_2)^{-L}, \\ a^{t_1} &\geq a^{t_2}, \end{aligned}$$

where $q > 1$, $L > 3$, and $a > 1$; therefore, Lemma 4.1 can then be applied and the theorem proved.

∎

The significance of Lemma 4.1 is that it ensures that one can adopt any monotonically nondecreasing function, and the resulting correlation associative memory will always be asymptotically stable. This proves to be very helpful when it comes to hardware implementation of correlation associative memories, because any physical device exhibits some deviation from its ideal response characteristic, especially when the input is outside its operating range. Consequently, if the real response is monotonically nondecreasing, then the system will always be stable, though the performance in storage capacity and error correction ability might become poorer. However, Lemma 4.1 only gives the sufficient condition for a CAM to be stable but says nothing about the necessary condition.

4.4 The Capacity and the Attraction Radius of ECAM

Since we feel that the exponential correlation associative memory is most suitable for VLSI implementation, this section is devoted to exploring the storage capacity and error correction capability of ECAM. Our definition of the storage capacity is similar to that of McEliece *et al.*'s [12]; i. e., if we choose $M = M(N)$ N-bit memory patterns at random, program an ECAM with those M memory patterns and initialize that ECAM with an input pattern that is $r \equiv \rho N$ bits away from the nearest memory pattern, we ask what the greatest rate of growth $M(N)$ as $N \to \infty$ is so that after one iteration, the bit error probability (P_e) is less than $e^{-T}/\sqrt{4\pi T}$, where T is a fixed and large. By adjusting T, one can make a tradeoff between the bit error probability and the storage capacity of an ECAM.

To begin with, assume that all M N-bit memory patterns $\mathbf{u}^{(k)}, k = 1, 2, \ldots, M$ are randomly chosen; in other words, each bit in any of the M memory patterns is the outcome of a Bernoulli trial (-1 or $+1$). Now let us present the theorem about the storage capacity of ECAM.

Theorem 4.3 : In an ECAM, if the initial state pattern $\mathbf{x}$ is ρN bits away from the nearest memory pattern and

$$M(N) = \begin{cases} \left(\frac{a^4}{4T}\right)\left(\frac{2}{(1+a^{-2})a^{2\rho'}}\right)^N + 1 & \text{if } \rho' < \frac{1}{1+a^2} \\ \left(\frac{a^4}{4T}\right) 2^{N(1-\mathcal{H}(\rho'))} + 1 & \text{if } \rho' \geq \frac{1}{1+a^2} \end{cases} \tag{4.6}$$

memory patterns are stored, where $\rho' = \rho + 1/N$ and $\mathcal{H}(\rho')$ is the binary information entropy of ρ', then as $N \to \infty$, the bit error probability, the probability that a bit in the next state pattern is not the same as the corresponding bit in the nearest memory pattern, is asymptotically less than $e^{-T}/\sqrt{4\pi T}$.

Proof : We will give only an outline of the proof, and details of the proof can be found in Appendix 4.B. To begin with, suppose the input $\mathbf{x}$ is $r = \rho N$ bits away from the nearest memory pattern, say $\mathbf{u}^{(l)}$; then the evolution equation becomes

$$\mathbf{x}' = sgn\left\{\sum_{k=1}^{M} a^{<\mathbf{u}^{(k)},\, \mathbf{x}>}\ \mathbf{u}^{(k)}\right\},$$

$$= \; sgn \left\{ a^{N(1-2\rho)} \, \mathbf{u}^{(l)} + \sum_{k=1,k\neq l}^{M} a^{<\mathbf{u}^{(k)},\, \mathbf{x}>} \, \mathbf{u}^{(k)} \right\}.$$

Considering only the i^{th} component of $\mathbf{x}'$ and letting $u_i^{(l)} = -1$ without loss of generality yield

$$x_i' \; = \; sgn \left\{ -a^{N(1-2\rho)} + \sum_{k=1,k\neq l}^{M} a^{<\mathbf{u}^{(k)},\, \mathbf{x}>} \, u_i^{(k)} \right\}. \qquad (4.7)$$

Note that the second term of the argument of the *sgn* function in the previous equation is a sum of $M-1$ i.i.d. (independent, identically-distributed) random variables. Define

$$\omega_k \; \equiv \; a^{<\mathbf{u}^{(k)},\, \mathbf{x}>} \, u_i^{(k)}, \qquad\qquad k = 1, 2, \ldots, M,$$

and let

$$\begin{aligned} \omega \; &\equiv \; \sum_{k=1,k\neq l}^{M} \omega_k, \\ v \; &\equiv \; \sum_{k=1}^{M} \omega_k \; = \; -a^{N(1-2\rho)} + \omega. \end{aligned}$$

After some lengthy derivation (see Appendix 4.B), we have the following results in order:

$$E[\omega] \; \ll \; a^{N(1-2\rho)}$$

and

$$Var[\omega] \; < \; \left(\frac{1}{2T}\right) \cdot a^{2N(1-2\rho)}.$$

As $N \to \infty$, $M \to \infty$, the central limit theorem [7] can be applied, which leads to

$$P_e \; < \; \frac{e^{-T}}{\sqrt{4\pi T}} \, . \qquad (4.8)$$

■

Thus we conclude that ECAM has a storage capacity that scales *exponentially* with N—the number of bits in each memory pattern. More importantly, in the case when $\rho' \geq \dfrac{1}{1+a^2}$, the storage capacity actually meets the ultimate sphere-packing bound in information theory [3]. However, note that from Figure 4.1, one sees that in order to store M memory patterns, one needs $M \times N$ connection weights and M exponentiation circuits and N hard-limiter neurons. Therefore, to store an exponential number of memory patterns, exponential hardware complexity is necessary.

This exponential capacity is very attractive; however, the dynamic range required of the exponentiation circuit grows exponentially with N. In any real implementation, this requirement is very difficult to meet, if not impossible. In a typical CMOS VLSI process, a transistor operating in the subthreshold region as an exponentiation circuit has a dynamic range of approximately 10^5 to 10^7 [8]. Hence, we need to study how the storage capacity of ECAM changes if the dynamic range of its exponentiation circuits is fixed.

Suppose the dynamic range (D) of the exponentiation circuits is fixed and

$$D \equiv a^N;$$

then as N increases, a will decrease, and M will no longer scale *exponentially* with N. We now concentrate on the case when N approaches infinity. Since N is very large and D is fixed, a will be near unity. Let

$$a \equiv 1 + \mu,$$

where μ is a small positive number; then

$$\log D = N \log a = N \log(1+\mu) \simeq N\mu.$$

As a becomes closer to unity, ρ' will be less than $1/(1+a^2)$ in practically all cases; therefore, only the first formula in Equation (4.6) needs to be considered. It follows that

$$M(N) = \left(\frac{a^4}{4T}\right)\left(\frac{2}{(1+a^{-2})a^{2\rho'}}\right)^N$$

$$\begin{aligned}
&\simeq \left(\frac{a^4}{4T}\right)\left(\frac{2}{(2-2\mu)(1+2\rho'\mu)}\right)^N \\
&\simeq \left(\frac{a^4}{4T}\right)\left(1+\frac{N\mu(1-2\rho')}{N}\right)^N \\
&\simeq \left(\frac{a^4}{4T}\right)\, e^{(1-2\rho)\log D} \simeq \left(\frac{a^4}{4T}\right)\, D^{1-2\rho}. \qquad (4.9)
\end{aligned}$$

4.5 Simulation Results

A few simulations have been conducted in order to confirm the theoretical results obtained. We let $a = 2$ and randomly choose 10 sets of M N-bit memory patterns in each case, then program an ECAM with these M patterns. For each ECAM, 100 input patterns are generated by randomly picking a memory pattern and flipping b bits. They are then fed to the ECAM and the ECAM is allowed to run until it becomes stable. The resulting fixed point is then compared with the original memory pattern, and the run is called a *success* if they match. We then collect the number of successes out of 1000 runs and if this number is greater than 998, we say that loaded with M memory patterns, ECAM can tolerate b errors. The largest b for a fixed M is called the *attraction radius* (r).

In Figure 4.2, the relative attraction radius $\rho \equiv r/N$ is plotted against the number of memory patterns (M) for various N. Note that if a horizontal line is drawn across the plot, it will intersect these four curves in the figure at points that are equally apart. Since the four curves correspond to cases with N that increases linearly, the previous observation implies that for fixed ρ the storage capacity of ECAM scales exponentially with N, which confirms Theorem 4.3. Next we let N be fixed and vary the dynamic range of exponentiation circuits. Figure 4.3 and Figure 4.4 illustrate how the the relationship between attraction radius (r) and the number of loaded memory patterns (M) changes for different dynamic ranges. As one can see, in both $N = 32$ and $N = 64$ cases, the curves intersect with the vertical axis ($r = 1$) at points that are approximately twice al large as the previous point. Since the dynamic ranges of these four curves increase by twofold successively, thus the storage capacity of ECAM is proportional to the dynamic range of the exponentiation circuits. Furthermore, if one again draws a vertical line with larger r, it will intersect the four curves at points equally apart, but this time the distance among these four points will be smaller than that of the case when $r = 1$. Therefore, we conclude that the previous result about the storage capacity of ECAM with fixed dynamic range exponentiation circuits; i. e., Equation (4.9) is valid.

4.6 Conclusions

In this chapter, we have discussed a group of associative memories that rely on the generalized correlation measure. We also prove that correlation associative memories based on the ordinary correlation are asymptotically stable as long as their weighting functions are monotonically nondecreasing. In particular, a new type of high-capacity neural network associative memory, which we call Exponential Correlation Associative Memory (ECAM), is presented. We have also investigated the storage capacity of ECAM under different assumptions, and find that under some condition ECAM meets the sphere-packing bound. Moreover, ECAM is more robust than the associative memory using the winner-take-all function to find the maximum correlation. In the latter associative memory, the final answer will be wrong if the winner-take-all function make a mistake. For the ECAM, the exact answer need not to be obtained at once. Since even if several out of the N binary decisions are wrong, it is still possible to reach a correct answer through iteration. Finally, we give simulation results showing that theoretical findings also apply to cases when N is not very large. VLSI implementation of ECAM and its application to some associative recall problems will be discussed in the next chapter.

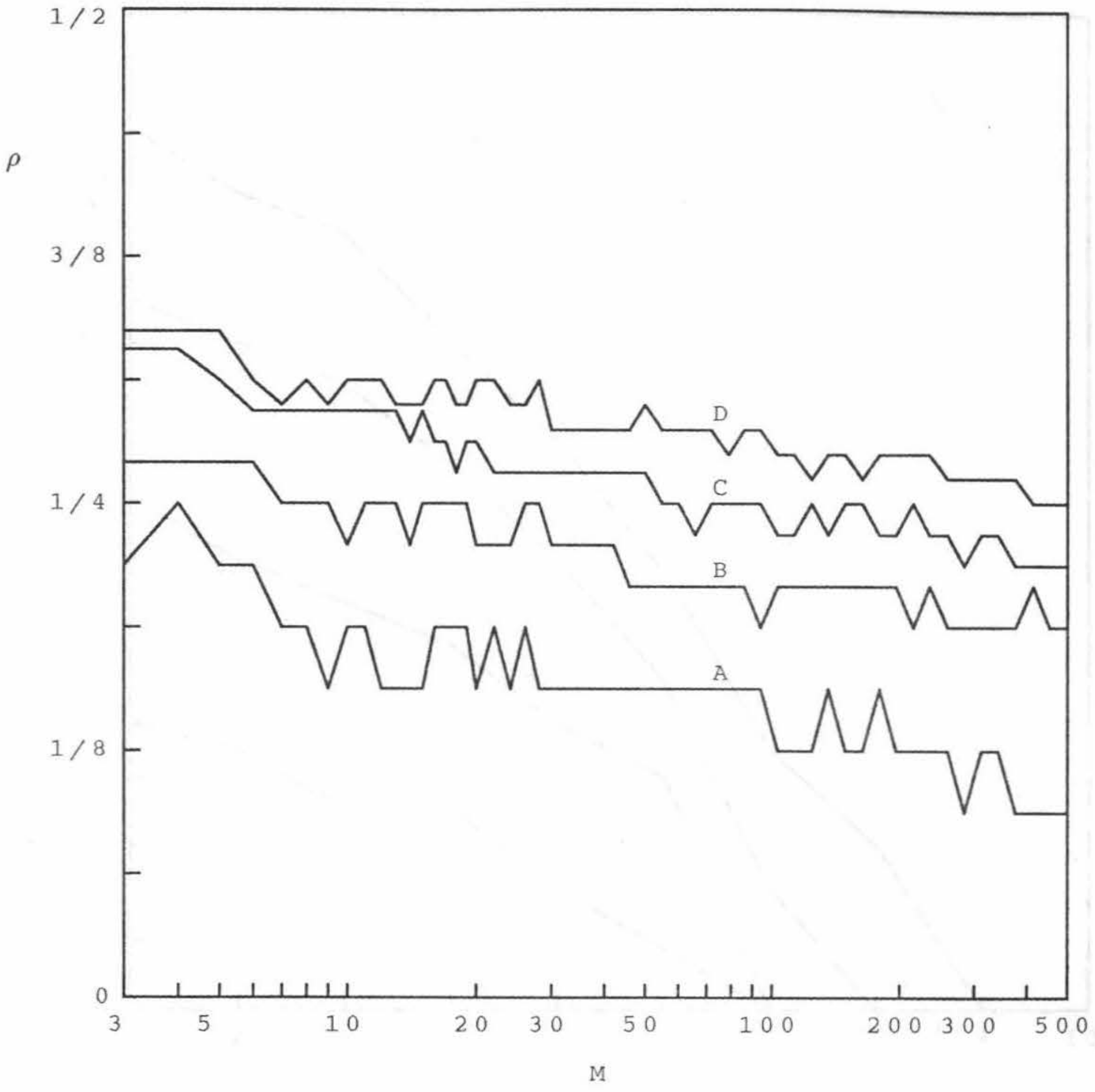


Figure 4.2: Attraction radius ($\rho = r/N$) vs. number of loaded memory patterns (M), Curve A : $N = 32$, Curve B : $N = 48$, Curve C : $N = 64$, Curve D : $N = 80$

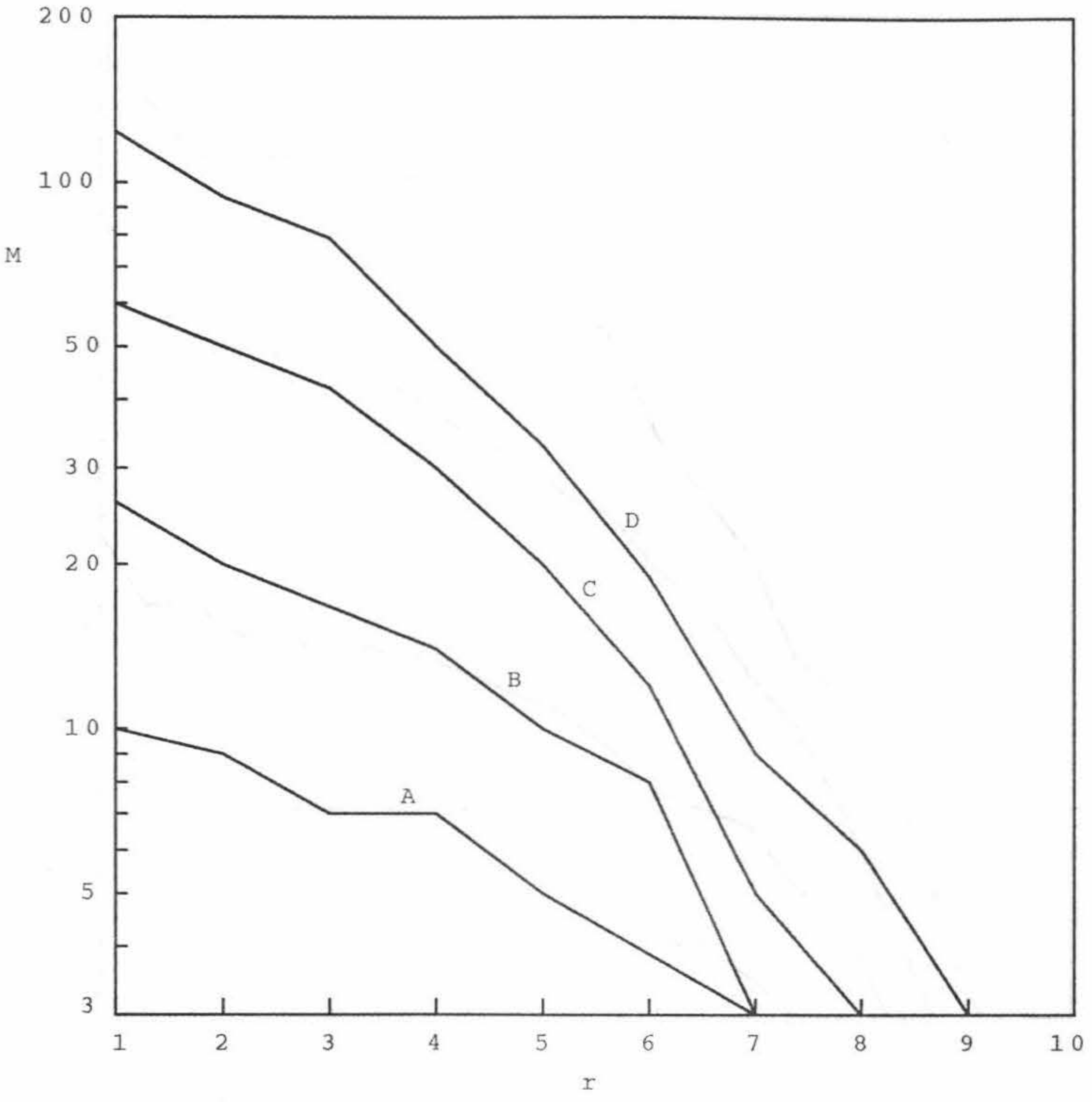


Figure 4.3: Number of loaded memory patterns (M) vs. attraction radius (r) with $N = 32$, Curve A : $D = 2^4$, Curve B : $D = 2^5$, Curve C : $D = 2^6$, Curve D : $D = 2^7$

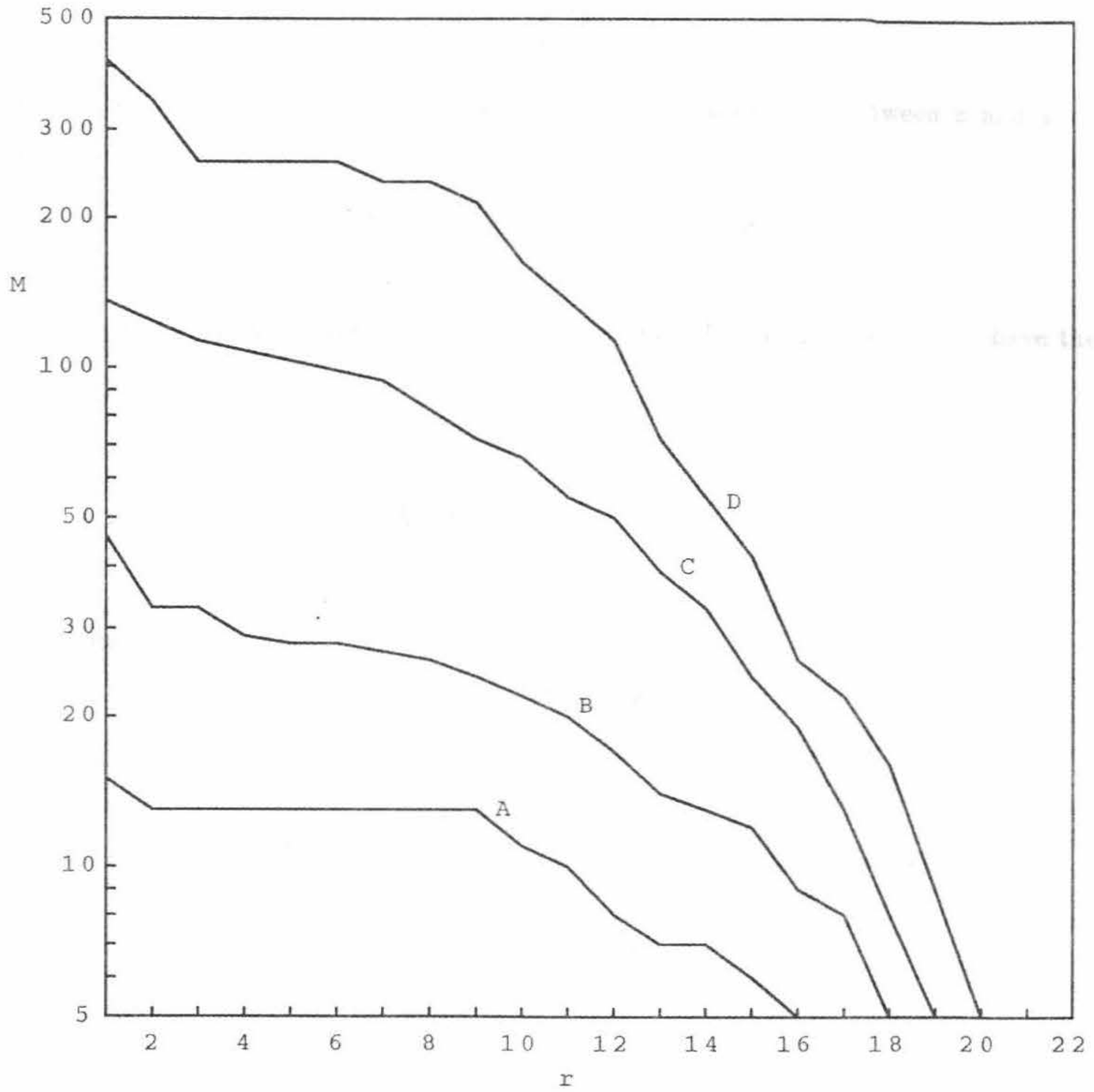


Figure 4.4: Number of loaded memory patterns (M) vs. attraction radius (r) with $N = 64$, Curve A : $D = 2^4$, Curve B : $D = 2^5$, Curve C : $D = 2^6$, Curve D : $D = 2^7$

Appendix 4.A Proof of the Convergence of CAMs

The proof of Lemma 4.1 is given in this appendix. At first, define

$$g(x) \equiv \int^{x} f(t)\, dt;$$

then by the mean value theorem, there exists some z, which lies between x and y, so that

$$g(y) - g(x) = g'(z) \cdot (y - x) = f(z) \cdot (y - x).$$

Next, by the assumption that $f(t)$ is monotonically nondecreasing, we have the following :

(a) $y > x$. In this case, we have

$$x \leq z \leq y,$$

thus

$$f(z) \geq f(x)$$

and

$$g(y) - g(x) \geq f(x) \cdot (y - x).$$

(b) $x > y$. In this case, we have

$$y \leq z \leq x,$$

thus

$$f(z) \leq f(x)$$

and

$$g(y) - g(x) \geq f(x) \cdot (y - x).$$

In both cases, we have the same inequality, $g(y) - g(x) \geq f(x) \cdot (y - x)$.

Now let us define an energy function for the correlation associative memory,

$$E(\mathbf{x}) \equiv -\sum_{k=1}^{M} g(<\mathbf{u}^{(k)}, \mathbf{x}>).$$

The difference between the energies of the current state and the state after all neurons perform one iteration according to the evolution equation is

$$\begin{aligned} -\Delta E &= -E(\mathbf{x}') + E(\mathbf{x}) \\ &= \sum_{k=1}^{M} g(<\mathbf{u}^{(k)}, \mathbf{x}'>) - g(<\mathbf{u}^{(k)}, \mathbf{x}>) \\ &\geq \sum_{k=1}^{M} g'(<\mathbf{u}^{(k)}, \mathbf{x}>) \cdot <\mathbf{u}^{(k)}, \mathbf{x}' - \mathbf{x}> \\ &= \sum_{k=1}^{M} f(<\mathbf{u}^{(k)}, \mathbf{x}>) \cdot \sum_{i=1}^{N} u_i^{(k)} \cdot (x_i' - x_i) \\ &= \sum_{i=1}^{N} \left\{ \sum_{k=1}^{M} f(<\mathbf{u}^{(k)}, \mathbf{x}>)\, c_i^{(k)} \right\} \cdot (x_i' - x_i) \\ &\geq 0. \end{aligned}$$

Furthermore, $\Delta E = 0$ only if all $x_i' - x_i = 0$ or 2; therefore, the associative memory will eventually become stable at a fixed point. By the same token, if only one neuron updates itself in every iteration, the associative memory will also converge to a fixed point.

∎

Appendix 4.B Proof of the Capacity Results of ECAM

In this appendix, a rigorous proof of Theorem 4.3 is presented.

For a given ρ, $0 \leq \rho < 1/2$, suppose the system is initialized with a state $\mathbf{x}$ which is $\rho N \equiv r$ bits away from the nearest memory pattern, say $\mathbf{u}^{(l)}$; in other words,

$$\mathbf{x} = \mathbf{u}^{(l)} + \mathbf{e},$$

where $\mathbf{e}$ has r nonzero ($+2$ or -2) components. Furthermore assume, without loss of generality, that $u_i^{(l)} = -1$. Since the bit error probability is larger for the case when $e_i = +2$ than when $e_i = 0$, only the former case will be studied.

First of all, since $\mathbf{u}^{(l)}$ is the nearest memory pattern to $\mathbf{x}$, all other $M-1$ memory patterns must be at least $r+1$ bits away from $\mathbf{x}$. Suppose

$$r' \equiv r+1, \text{ and } \rho' \equiv r'/N = \rho + 1/N;$$

we then calculate the probability distribution function of the random variable ω_1 when $e_i = +2$; i. e., $x_i = +1$ as

$$\begin{aligned}
\text{Prob}[\,\omega_1 = a^{N-2j}\,] &= \frac{1}{K}\binom{N-1}{j}, & j &= r', r'+1, \ldots, N-1. \\
\text{Prob}[\omega_1 = -a^{N-2j-2}] &= \frac{1}{K}\binom{N-1}{j}, & j &= r'-1, r', \ldots, N-1.
\end{aligned}$$

The first formula applies to the case when $u_i^{(1)} = +1$, while the second applies to the case when $u_i^{(1)} = -1$. The constant K is a normalizing factor and

$$\begin{aligned}
K &= \sum_{j=r'}^{N-1}\binom{N-1}{j} + \sum_{j=r'-1}^{N-1}\binom{N-1}{j} \\
&= \sum_{j=r'}^{N-1}\left[\binom{N-1}{j} + \binom{N-1}{j-1}\right] + \binom{N-1}{N-1}
\end{aligned}$$

$$= \sum_{j=r'}^{N-1} \binom{N}{j} + \binom{N}{N} = \sum_{j=r'}^{N} \binom{N}{j}.$$

Also, since $\rho < 1/2$, then $r' = \rho N + 1 \leq \lceil N/2 \rceil$, and

$$K \ \geq \ \sum_{j=\lceil N/2 \rceil}^{N} \binom{N}{j} \ \geq \ 2^{N-1}. \tag{4.10}$$

Next let us calculate the expectation of ω_1,

$$\begin{aligned} E[\omega_1] &= \frac{1}{K}\left\{\sum_{j=r'}^{N-1}\binom{N-1}{j} a^{N-2j} - \sum_{j=r'-1}^{N-1}\binom{N-1}{j} a^{N-2j-2}\right\} \\ &< \frac{a^N}{2^{N-1}}\left\{\sum_{j=r'}^{N-1}\binom{N-1}{j} a^{-2j}\right\} \\ &< \frac{a^N}{2^{N-1}}\left\{\sum_{j=r'}^{N}\binom{N}{j} a^{-2j}\right\}. \end{aligned}$$

Obviously, $E[\omega_1]$ is positive. Now in order to bound it from above, we apply the Chernoff method. Multiplying each term in the summation by a number greater than or equal to unity ($e^{t(j-r')}$, $t \geq 0$) and summing from $j = 0$ instead of from $j = r'$ gives

$$\begin{aligned} E[\omega_1] &< \frac{a^N}{2^{N-1}}\left\{\sum_{j=r'}^{N}\binom{N}{j} a^{-2j} e^{t(j-r')}\right\} \\ &< \left(\frac{a^N}{2^{N-1}}\right) e^{-tr'}\left\{\sum_{j=0}^{N}\binom{N}{j}\left(a^{-2}e^{t}\right)^{j}\right\} \\ &= \left(\frac{a^N}{2^{N-1}}\right) e^{-tr'}\left(1 + a^{-2}e^{t}\right)^{N}, \qquad \text{where } t \geq 0. \end{aligned}$$

Similarly, the expectation of ω_1^2 can be bounded, and we have

$$E[\omega_1^2] = \frac{1}{K}\left\{\sum_{j=r'}^{N-1}\binom{N-1}{j} a^{2N-4j} + \sum_{j=r'-1}^{N-1}\binom{N-1}{j} a^{2N-4j-4}\right\}$$

$$
\begin{aligned}
&< \frac{a^{2N}}{K}\left\{\sum_{j=r'}^{N-1}\binom{N-1}{j}a^{-4j}\right\} \\
&< \frac{a^{2N}}{2^{N-1}}\left\{\sum_{j=r'}^{N}\binom{N}{j}a^{-4j}\right\} \\
&< \frac{a^{2N}}{2^{N-1}}\left\{\sum_{j=0}^{N}\binom{N}{j}a^{-4j}e^{t(j-r')}\right\} \\
&= \left(\frac{a^{2N}}{2^{N-1}}\right)e^{-tr'}\left(1+a^{-4}e^{t}\right)^{N}, \qquad \text{where } t \geq 0.
\end{aligned}
$$

Accordingly, the variance of ω_1 is

$$
\begin{aligned}
Var[\omega_1] &= E[\omega_1^2] - E[\omega_1]^2 \\
&\leq E[\omega_1^2] \\
&< \left(\frac{a^{2N}}{2^{N-1}}\right)e^{-tr'}\left(1+a^{-4}e^{t}\right)^{N}, \qquad \text{where } t \geq 0.
\end{aligned}
$$

Since ω is the sum of $M-1$ i.i.d. random variables, the expectation and the variance of ω are both $M-1$ times those of ω_1; namely,

$$
\begin{aligned}
E[\omega] &= (M-1)\,E[\omega_1] \\
&< (M-1)\left(\frac{a^{N}}{2^{N-1}}\right)e^{-tr'}\left(1+a^{-2}e^{t}\right)^{N}, \qquad \text{where } t \geq 0. \qquad (4.11)
\end{aligned}
$$

$$
\begin{aligned}
Var[\omega] &= (M-1)\,Var[\omega_1] \\
&< (M-1)\left(\frac{a^{2N}}{2^{N-1}}\right)e^{-tr'}\left(1+a^{-4}e^{t}\right)^{N}, \qquad \text{where } t \geq 0. \qquad (4.12)
\end{aligned}
$$

To estimate the bit error probability, we need to deal with two cases separately.

(a) $\rho' \geq \dfrac{1}{1+a^2}$

Since Equations (4.11) and (4.12) are true for all nonnegative t, we can find an optimal t so that the right hand sides of both equations are minimized. In Equation (4.11), let

$$e^t = \frac{a^2\rho'}{1-\rho'} \geq \frac{a^2(1+a^2)^{-1}}{1-(1+a^2)^{-1}} = 1.$$

then

$$\begin{aligned} E[\omega] &< (M-1)\left(\frac{a^N}{2^{N-1}}\right)\left(\frac{1-\rho'}{a^2\rho'}\right)^{\rho' N}\left(\frac{1}{1-\rho'}\right)^N \\ &= (M-1)\left(\frac{a^{N(1-2\rho')}}{2^{N-1}}\right)\left(\frac{1}{\rho'}\right)^{\rho' N}\left(\frac{1}{1-\rho'}\right)^{(1-\rho')N} \\ &= 2(M-1)\cdot a^{N(1-2\rho')}\ .\ 2^{N(\mathcal{H}(\rho')-1)}, \end{aligned}$$

where $\mathcal{H}(x) = -x\log_2 x - (1-x)\log_2(1-x)$, the binary information entropy of ρ'.

Assume that T is large and let

$$M(N) = \left(\frac{a^4}{4T}\right)2^{N(1-\mathcal{H}(\rho'))} + 1; \tag{4.13}$$

it then follows that

$$E[\omega] < \left(\frac{a^2}{2T}\right)a^{N(1-2\rho)} \ll a^{N(1-2\rho)}. \tag{4.14}$$

Similarly, the variance of ω can also be upper bounded by substituting

$$e^t = \frac{a^4\rho'}{1-\rho'} \geq a^2 > 1$$

in Equation (4.12).

$$\begin{aligned} Var[\omega] &< (M-1)\left(\frac{a^{2N(1-2\rho')}}{2^{N-1}}\right)\left(\frac{1}{\rho'}\right)^{\rho' N}\left(\frac{1}{1-\rho'}\right)^{(1-\rho')N} \\ &= 2(M-1)\cdot a^{2N(1-2\rho')}\ .\ 2^{N(\mathcal{H}(\rho')-1)} \end{aligned}$$

By Equation (4.13) and the above equation, we obtain

$$Var[\omega] \;<\; \left(\frac{1}{2T}\right) \cdot a^{2N(1-2\rho)}. \tag{4.15}$$

(b) $\rho' < \dfrac{1}{1+a^2}$

Substituting $e^t = 1$ in Equation (4.11), we have

$$\begin{aligned} E[\omega] \;&<\; (M-1)\left(\frac{a^N}{2^{N-1}}\right)\left(1+a^{-2}\right)^N \\ &=\; 2(M-1)\cdot a^{N(1-2\rho')}\cdot\left(\frac{(1+a^{-2})a^{2\rho'}}{2}\right)^N. \end{aligned}$$

Now suppose that T is large and

$$M(N) \;=\; \frac{a^4}{4T}\left(\frac{2}{(1+a^{-2})a^{2\rho'}}\right)^N + 1; \tag{4.16}$$

then

$$E[\omega] \;<\; \left(\frac{a^2}{2T}\right) a^{N(1-2\rho)} \;\ll\; a^{N(1-2\rho)}. \tag{4.17}$$

Next, an upper bound of $Var[\omega]$ can be calculated by setting $e^t = a^2$ in Equation (4.12),

$$\begin{aligned} Var[\omega] \;&<\; (M-1)\left(\frac{a^{2N(1-\rho')}}{2^{N-1}}\right)\left(1+a^{-2}\right)^N \\ &=\; 2(M-1)\cdot a^{2N(1-2\rho')}\cdot\left(\frac{(1+a^{-2})a^{2\rho'}}{2}\right)^N. \end{aligned}$$

Combining Equation (4.16) and the above equation leads to

$$Var[\omega] \;<\; \left(\frac{1}{2T}\right) \cdot a^{2N(1-2\rho)}. \tag{4.18}$$

We have shown in both cases that $E[\omega]$ is significantly smaller than $a^{N(1-2\rho)}$ when T is large and thus can be ignored. Also $Var[\omega]$ is found to be bounded above by the same quantity in both cases. We now estimate the probability that an ECAM is not able to correct a bit error, namely, the probability that $v > 0$. Since the random variable ω is the sum of $M-1$ i.i.d. random variables, as $N, M \to \infty$, ω can be approximated by a normal distribution, i. e., $\mathcal{N}(E[\omega], Var[\omega])$ (the central limit theorem [7]). Therefore,

$$\begin{aligned}
\text{Prob}[v > 0] &= \text{Prob}[\ \omega > a^{N(1-2\rho)}\ ] \\
&= \text{Prob}[\ \frac{\omega - E[\omega]}{\sigma_\omega} > \frac{a^{N(1-2\rho)} - E[\omega]}{\sigma_\omega}\ ] \\
&\simeq \text{Prob}[\ \frac{\omega - E[\omega]}{\sigma_\omega} > \frac{a^{N(1-2\rho)}}{\sigma_\omega}\ ] \\
&= Q\left(\sqrt{\frac{a^{2N(1-2\rho)}}{Var[\omega]}}\right) \ < \ Q\left(\sqrt{2T}\right),
\end{aligned} \tag{4.19}$$

where σ_ω is the standard deviation of ω and

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-t^2/2}\, dt.$$

Note that T is fixed, so we do not have to worry about the large deviation problem in applying the central limit theorem. If T is large, we can use the asymptotic formula for $Q(\cdot)$:

$$Q(t) \simeq \frac{1}{(\sqrt{2\pi}\,)\,t} \cdot e^{-t^2/2}.$$

By the above formula and Equation (4.19), we have

$$\begin{aligned}
P_e &= \text{Prob}[v > 0] \\
&< \frac{e^{-T}}{\sqrt{4\pi T}}.
\end{aligned} \tag{4.20}$$

■

References

[1] J. Bruck and J. W. Goodman, "A Generalized Convergence Theorem for Neural Networks and Its Applications in Combinatorial Optimization," in *Proc. Int. Conf. on Neural Networks*, San Diego, CA, Vol. III, 1987, pp. 649–656.

[2] T. D. Chiueh and R. M. Goodman, "High-Capacity Exponential Associative Memory," in *Proc. Int. Conf. on Neural Networks*, San Diego, CA, Vol. I, 1988, pp. 153–160.

[3] P. A. Chou, "The Capacity of the Kanerva Associative Memory Is Exponential," in *Neural Information Processing Systems*, D. Z. Anderson (editor), New York, NY : American Institute of Physics, 1988, pp. 184–191.

[4] A. Dembo and O. Zeitouni, "High Density Associative Memories," in *Neural Information Processing Systems*, D. Z. Anderson (editor), New York, NY : American Institute of Physics, 1988, pp. 211–218.

[5] A. Dembo and O. Zeitouni, "General Potential Surfaces and Neural Networks," *Physical Review A*, Vol. 37, No. 6, 2134–2143, 1988.

[6] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis.* New York, NY : John Wiley and Sons, 1973.

[7] W. Feller, *An Introduction to Probability Theory and its Application*, Vol. II, 2nd edition. New York, NY : John Wiley and Sons, 1971.

[8] L. A. Glasser and D. W. Dopperpuhl, *The Design and Analysis of VLSI Circuits.* Reading, MA : Addison-Wesley, 1985.

[9] J. J. Hopfield, "Neural Network and Physical Systems with Emergent Collective Computational Abilities," *Proc. Nat. Acad. Sci. USA*, Vol. 79, 2554–2558, 1982.

[10] J. J. Hopfield, "Neurons with Graded Response Have Collective Computational Properties Like Those of Two-State Neurons," *Proc. Nat. Acad. Sci. USA*, Vol. 81, 3088–3092, 1984.

[11] Y. C. Lee, *et al.*, "Machine Learning Using a Higher Order Correlation Network," *Physica 22D*, 276–306, 1986.

[12] R. J. McEliece, E. C. Posner, E. R. Rodemich, S. S. Venkatesh, "The Capacity of The Hopfield Associative Memory," *IEEE Tran. on Information Theory*, Vol. IT-33, 461–482, 1987.

[13] D. Psaltis and C. H. Park, "Nonlinear Discriminant Functions and Associative Memory," in *Neural Networks for Computing*, J. S. Denker (editor), New York, NY : American Institute of Physics, 1986, pp. 370–375.

[14] M. R. Sayeh and J. Y. Han, "Pattern Recognition Using a Neural Network," in *Proc. of SPIE Cambridge Symp. on Opt. and Optoelec. Eng.*, Cambridge, MA, Nov., 1987.

[15] B. Soffer, "Holographic Associative Memories," in *Proc. of Workshop on Neural Network Devices and Applications*, Jet Propulsion Lab., Feb., 1987, pp. 125–146.

Chapter 5

VLSI Implementation of ECAM

5.1 Introduction

In the previous chapters, we introduced a model for correlation associative memories, which includes a variation of the Hopfield memory and the high-order associative memories as special cases. This model is based on an architecture consisting of binary connection weights, simple hard-limiter neurons, and specialized nonlinear circuits. The evolution equation of this general model is

$$\mathbf{x}' = sgn \left\{ \sum_{k=1}^{M} f(<\mathbf{u}^{(k)}, \mathbf{x}>)\, \mathbf{u}^{(k)} \right\}, \tag{5.1}$$

where $\mathbf{u}^{(1)}, \mathbf{u}^{(2)}, \cdots, \mathbf{u}^{(M)}$ are the M memory patterns. $\mathbf{x}$ and $\mathbf{x}'$ are the current and the next state patterns of the system, respectively, and sgn is the threshold function, which takes on the value $+1$ if its argument is nonnegative, and -1 otherwise.

We addressed, in particular, the case when f is of the exponentiation form, namely, when the evolution equation is

$$\mathbf{x}' = sgn \left\{ \sum_{k=1}^{M} a^{<\mathbf{u}^{(k)}, \mathbf{x}>}\ \mathbf{u}^{(k)} \right\}, \tag{5.2}$$

and a is a constant greater than unity. The new exponential correlation associative memory (ECAM) possesses a very large storage capacity, which scales *exponentially* with the length of memory patterns [1]. Furthermore, it has been shown that ECAM is asymptotically stable in both synchronous and asynchronous updating modes.

The ECAM chip we designed is *programmable*; that is, one can change the set of memory patterns of that associative memory at will. To perform an associative recall, one first loads a set of memory patterns onto the chip. The chip is then switched to the associative recall mode, and an input pattern is presented to the ECAM chip. The ECAM chip then computes the next state pattern according to Equation (5.2) and presents it at the output port of the chip. The components of the next state patterns appear in parallel, after the internal circuits have settled, at the output port. Feedback is easily incorporated by connecting the output port to the input port. In this case, the ECAM chip will cycle until a fixed point is reached.

In this chapter, we first describe a VLSI implementation of ECAM, present the testing results of the ECAM chip, compare the error correcting performance of the real chip and that of a simulated ECAM, and finally demonstrate the speed and capability of the ECAM chip by utilizing it in performing vector quantization on binary images.

5.2 Circuit Design of the Static RAM

Because of the need for remembering and changing memory patterns of the associative memory, the ECAM chip is built on top of a basic RAM structure. We first elucidate the RAM structure, and then describe associative recall circuits as well as input/output peripheral circuits. The static RAM consists of three major components, which will be described in the following subsections.

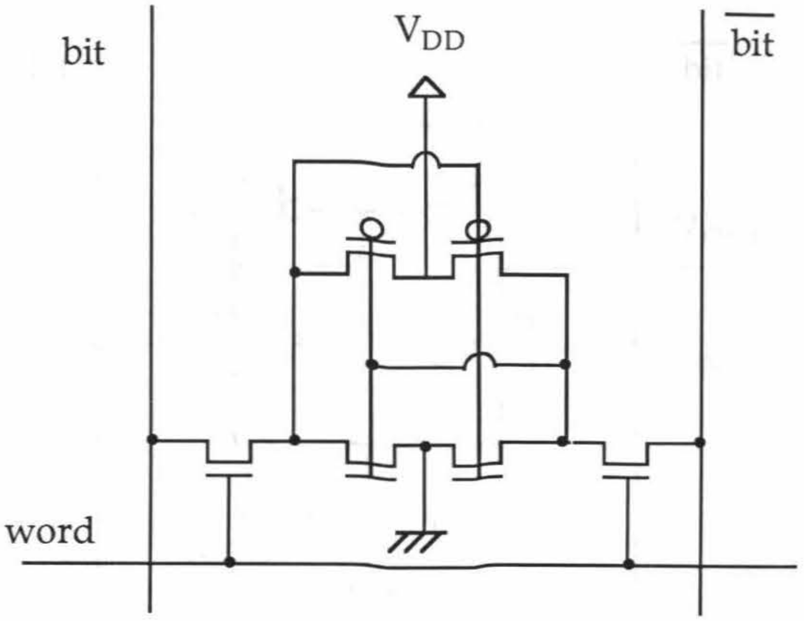


Figure 5.1: A six-transistor static random access memory cell

5.2.1 Memory Cell

A six-transistor static RAM cell is chosen as the memory cell used in the ECAM chip because it requires least peripheral control and because it is most reliable. The RAM cell consists of two coupled inverters holding one bit of information. This bit can be accessed by dual-rail bit lines through two NMOS transistors controlled by a word line (see Figure 5.1). If a cell is not being accessed, the word line that controls that cell remains Low; hence the cell is isolated from the two bit lines. Even with the presence of leakage current, information is preserved by the feedback configuration of a coupled inverter pair. Also note that both the cell's information bit and its complement are available to other circuits.

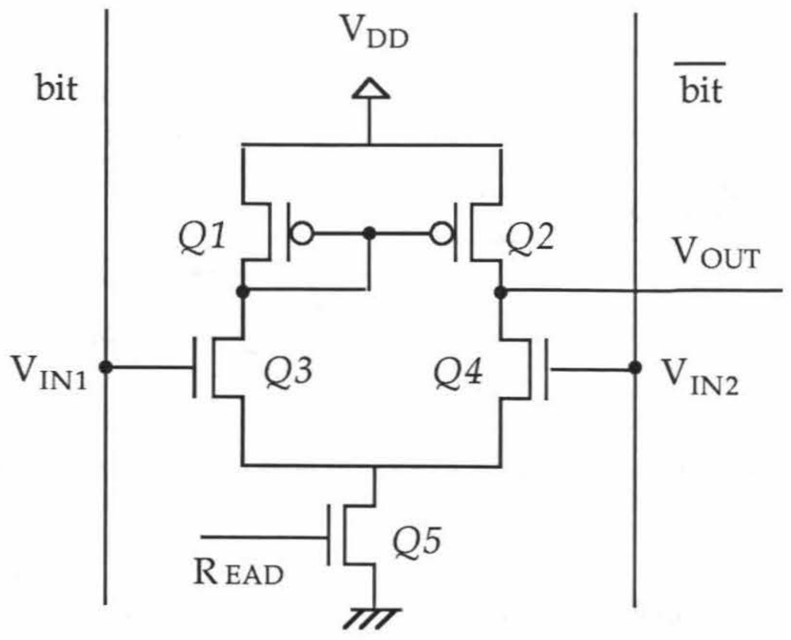


Figure 5.2: Circuit diagram of the sense amplifier used in the ECAM chip

5.2.2 Sense Amplifier

We choose a simple differential amplifier as the sense amplifier used in the ECAM chip [6, 2] (see Figure 5.2). This amplifier, albeit simple, has been shown, by SPICE simulation, to be good enough for this particular application .

In Figure 5.2, $Q5$ works as a current source activated by the signal "READ." If VIN1 is higher than VIN2, then the gate-to-source voltage of $Q3$ is larger than the gate-to-source voltage of $Q4$. Therefore, $Q3$ will sink more current than $Q4$. Moreover, the sum of the drain currents in $Q3$ and $Q4$ equals the drain current in $Q5$ and is fixed. Consequently, if VIN1 is sufficiently higher than VIN2, $Q3$ sinks a current orders of magnitude larger than $Q4$ does, and $Q4$ is effectively cut off. Since $Q1$ and $Q2$ have the same gate-to-source voltage, the drain-to-source voltage of $Q2$ must be near zero; i. e.,

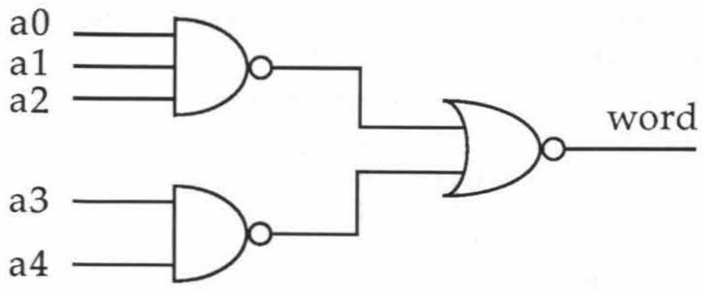


Figure 5.3: Circuit diagram of the row decoder used in the ECAM chip

VOUT will be near VDD. On the other hand, if VIN2 is sufficiently higher than VIN1, then the gate-to-source of $Q4$ is higher than that of $Q3$. Therefore, $Q4$ will sink most current that passes through the current source transistor $Q5$. Also, since $Q1$ and $Q2$ have the same gate-to-source voltage, the drain-to-source voltage of $Q2$ must be large and VOUT must be near GND. In conclusion, we see that if VIN1 is sufficiently higher than VIN2, then VOUT will be VDD; if the reverse is true, VOUT will be GND.

5.2.3 Row Decoder

We choose a static NAND-NOR type decoder for row decoding [7] because of its reliability and ease of implementation (see Figure 5.3). The associative memory chip is able to store 32 memory patterns addressed by five address bits (a0 – a4). The output of a row decoder goes to a word line driver, which provides the signal that controls the transmission gates in the memory cells.

word line driver size	bit line driver size	reading "1" writing "0" T_r(ns)	T_w(ns)	reading "0" writing "1" T_r(ns)	T_w(ns)
x4	x4	19.4	13.5	11.8	14.1
x4	x8	19.0	11.6	11.8	12.0
x4	x16	19.2	11.0	11.9	11.0
x8	x4	14.8	13.5	11.5	14.3
x8	x8	14.8	11.6	11.3	12.0
x8	x16	14.3	11.0	11.4	11.0
x16	x4	11.0	13.5	11.3	13.9
x16	x8	11.2	11.6	11.3	12.0
x16	x16	10.9	11.0	11.3	11.0

Table 5.1: Read time (T_r) and write time (T_w) of the static RAM with various driver sizes obtained from SPICE simulation

5.3 SPICE Simulation Results of the Static RAM

In order to determine the ratio of the sizes of global line drivers and transistors in memory cells, we conduct extensive SPICE simulation on relevant circuits. Table 5.1 illustrates the access times during reading and writing for various driver sizes and minimum size transistors in memory cells.

The first thing one notices is that the size of word line drivers does not affect the write access time (T_w) significantly in both cases. This is because in the simulation a read-modify-write cycle is simulated; therefore, the two transmission gates controlled by the word line are already closed before a write operation begins.

In a read operation, as soon as the signal "READ" rises, the current source transistor is turned on and starts sinking current. This pulls the output of the sense amplifier down before the amplifier even has a chance to sense the difference between the two bit lines.

current source transistor size	reading "1" writing "0"		reading "0" writing "1"	
	T_r(ns)	T_w(ns)	T_r(ns)	T_w(ns)
x1	14.8	11.6	11.5	12.1
x2	19.4	11.7	10.5	12.1
x3	20.2	11.8	10.1	12.1

Table 5.2: Read time (T_r) and write time (T_r) of the static RAM with various current source sizes obtained from SPICE simulation

Consequently, when reading a "0," the word line driver size has almost no influence on read access time (T_r). But when reading a "1," T_r decreases as the word line drivers get bigger because the bigger the word line drivers, the sooner the transmission gates close and the sooner the sense amplifiers produce valid output. As for the bit line drivers, making them bigger decreases T_w and at the same time increases T_r slightly. Bigger bit line drivers mean more load for the pull-up transistors in the memory cells, which in turn slows down the operation of the sense amplifiers and thus induces longer read access time.

We decide to balance the effects of these two parameters and choose a configuration that makes both T_r and T_w small enough without occupying too much silicon real estate. The transistors in the word line drivers and bit line drivers are made 8 times as wide as the minimum size transistor, i. e., 32λ wide and 4λ long in the scalable CMOS technology. Also, the pull down transistors in memory cells are of minimum size so that as many cells as possible can be put on the chip.

We also look into the effects of varying the size of the current source transistors in the sense amplifiers. Referring to Table 5.2, one notices that making current source transistors bigger does not produce any significant improvement. When the current source is made twice as strong, T_r decreases when reading "0" and increases when reading "1." This is because once the current source is turned on, the current flowing through the current source transistor brings the output of the sense amplifier low. Therefor, bigger current source means larger current, more voltage drop at the output, and longer time to read "1." We therefore decide to have all current source transistors of minimum size.

5.4 Design of Associative Recall Circuits

In this section, analog computation circuits implementing the ECAM evolution equation for associative recall are described. From the evolution equation of ECAM, we notice that there are essentially three circuits that need to be designed in order to build an ECAM chip. They are:

- $<\mathbf{u}^{(k)}, \mathbf{x}>$, the correlation computation circuit;
- $\sum_{k=1}^{M} a^{<\mathbf{u}^{(k)}, \mathbf{x}>}\ \mathbf{u}^{(k)}$, exponentiation, multiplication and summing circuit;
- $sgn(\cdot)$, the threshold circuit.

Now let us describe each circuit, present its design and simulation results, and finally integrate all these circuits to get the complete design of an ECAM chip.

5.4.1 Correlation Computation Circuit

In Figure 5.4, we illustrate a voltage-divider type circuit consisting of NMOS transistors working as controlled resistors (linear resistors or open circuits). This circuit computes the correlation between the input pattern $\mathbf{x}$ and a memory pattern $\mathbf{u}^{(k)}$. If the i^{th}

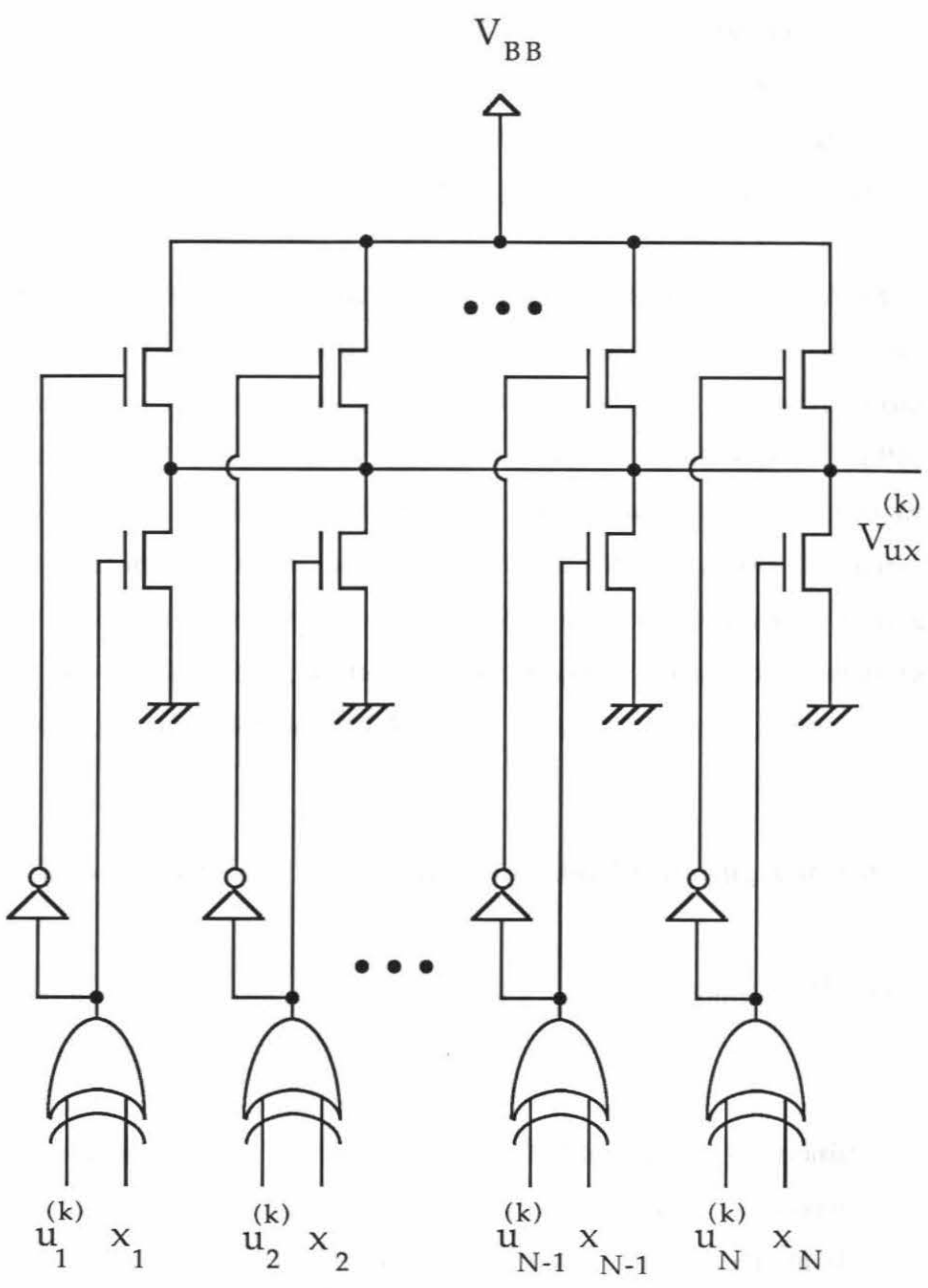


Figure 5.4: Circuit diagram of the correlation computation circuit in the ECAM chip

components of these two patterns are the same, the corresponding XOR gate outputs a "0" and there is a connection from the node $\mathrm{V}_{\mathrm{ux}}^{(\mathrm{k})}$ to VBB; otherwise, there is a connection from $\mathrm{V}_{\mathrm{ux}}^{(\mathrm{k})}$ to GND. Hence the output voltage will be proportional to the number of positions at which $\mathbf{x}$ and $\mathbf{u}^{(k)}$ match. The maximum output voltage is controlled by an externally supplied voltage VBB. Normally, VBB is set to a voltage lower than the threshold voltage of NMOS transistors (VTH) for a reason that will be explained later.

The conductance of an NMOS transistor in the ON mode is not fixed, but rather depends on its gate-to-source voltage and its drain-to-source voltage. Thus, some nonlinearity is due to occur in the correlation computation circuit. A correlation computation circuit with $N = 64$ is simulated by SPICE. In Figure 5.5, we illustrate the SPICE output voltage $\mathrm{V}_{\mathrm{ux}}^{(\mathrm{k})}$ and the ideal linear response, i. e., the case when ON transistors are replaced by linear resistors and OFF transistors by open circuits. As shown in Figure 5.5, there is only slight deviation from the ideal response throughout the whole operating range — from 0V to VBB. Therefore, we feel that the proposed correlation computation circuit should be good enough for the ECAM chip.

5.4.2 Exponentiation, Multiplication, and Summing Circuit

Figure 5.6 depicts a circuit that computes the exponentiation of $\mathrm{V}_{\mathrm{ux}}^{(\mathrm{k})}$, the product of $u_i^{(k)}$ and the exponential, and the sum of all M products.

The exponentiation function is implemented by an NMOS transistor whose gate voltage is $\mathrm{V}_{\mathrm{ux}}^{(\mathrm{k})}$. Since VBB, the maximum value that $\mathrm{V}_{\mathrm{ux}}^{(\mathrm{k})}$ can assume, is set to be lower than the threshold voltage (VTH); the NMOS transistor is in the subthreshold region, where its drain current depends exponentially on its gate-to-source voltage [5]. If we temporarily ignore transistors controlled by $u_i^{(k)}$ or the complement of $u_i^{(k)}$, the current flowing through the exponentiation transistor associated with $\mathrm{V}_{\mathrm{ux}}^{(\mathrm{k})}$ will scale exponentially with $\mathrm{V}_{\mathrm{ux}}^{(\mathrm{k})}$. Therefore, the exponentiation function is properly computed.

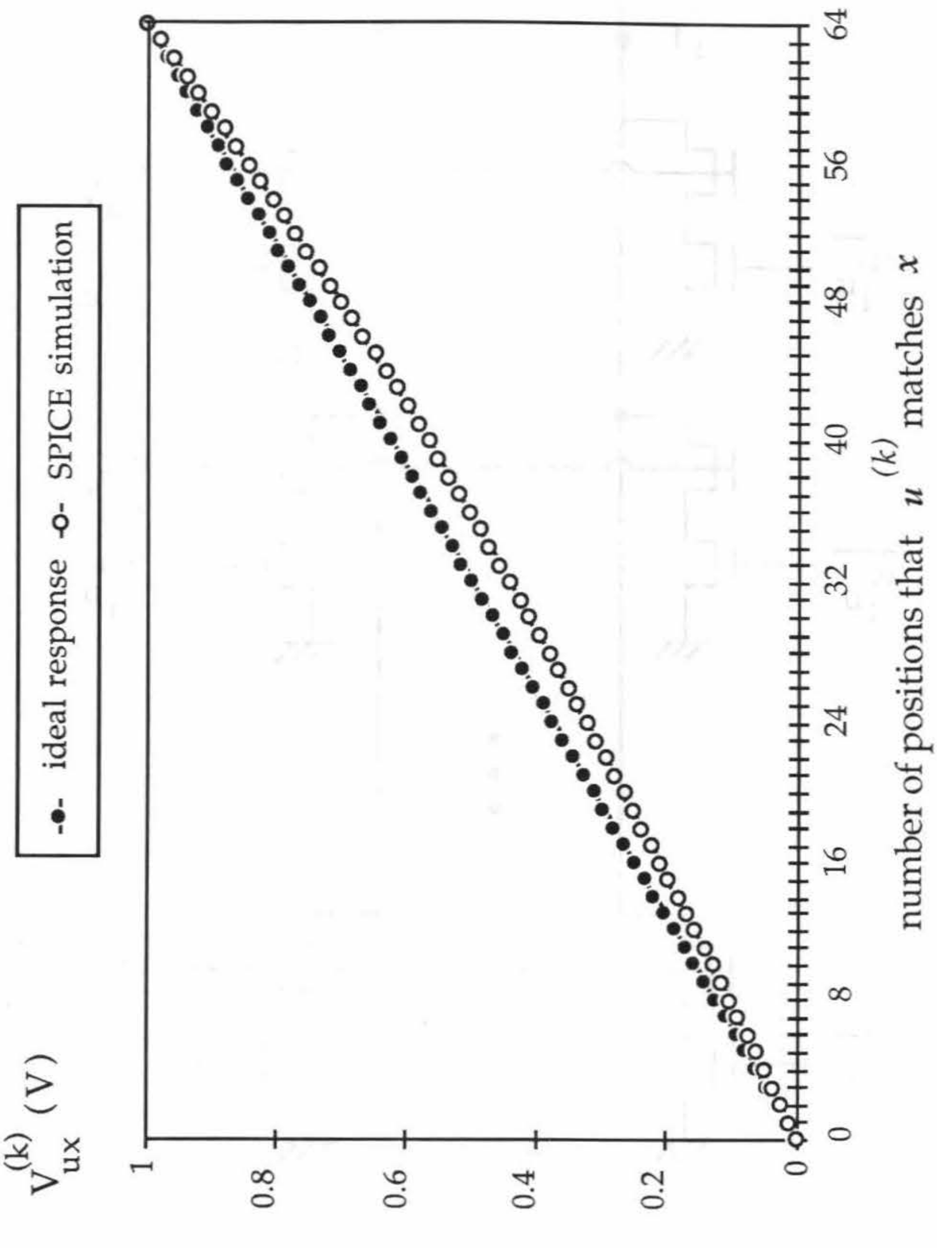


Figure 5.5: Comparison of the output voltage of a SPICE simulated correlation computation circuit with $N = 64$ and the ideal response

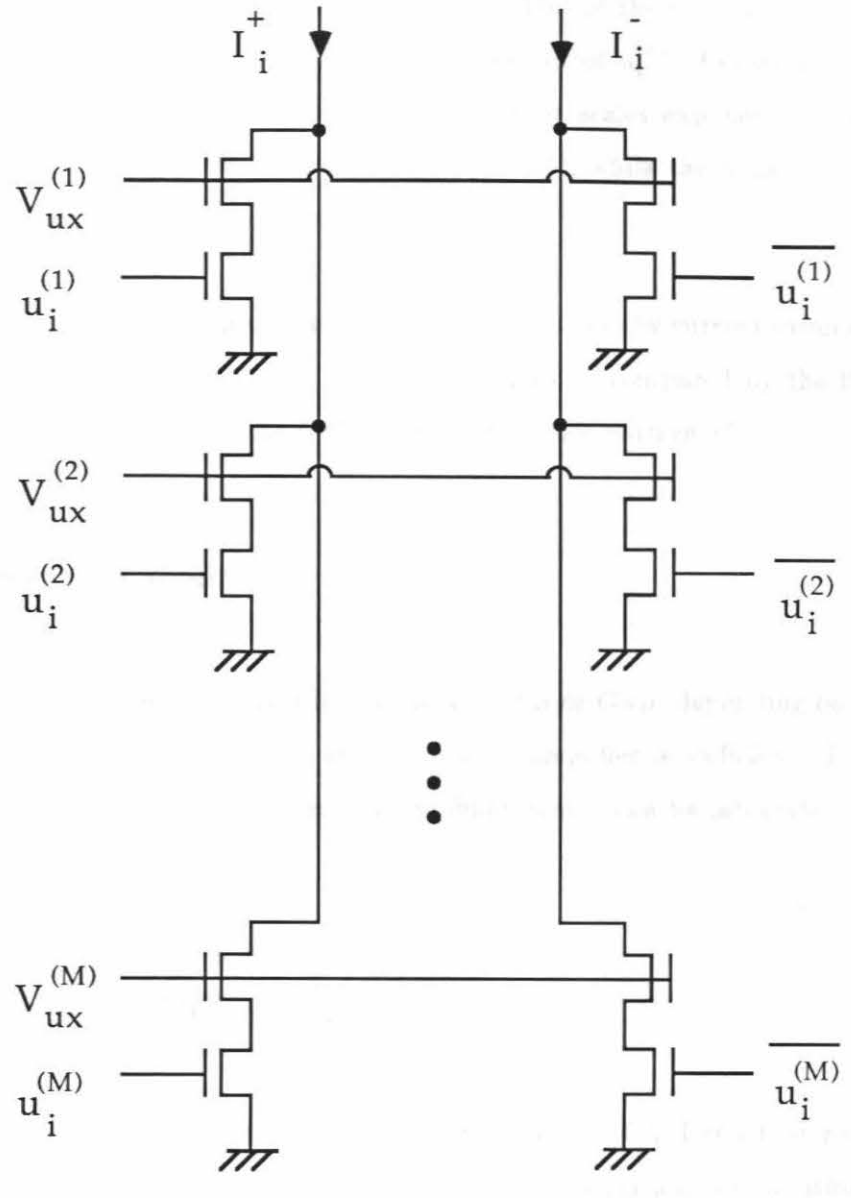


Figure 5.6: Circuit diagram of the exponentiation, multiplication, and summing circuit in the ECAM chip

Since the multiplier $u_i^{(k)}$ assumes either $+1$ or -1, the multiplication can be easily done by forming two branches, each made up of a transmission gate in series with an exponentiation transistor whose gate voltage is $\mathrm{V}_{\mathbf{ux}}^{(\mathbf{k})}$. One of the two transmission gates is controlled by $u_i^{(k)}$, and the other by the complement of $u_i^{(k)}$. Consequently, when $u_i^{(k)} = 1$, the positive branch will carry a current that scales exponentially with the correlation of input $\mathbf{x}$ and the k^{th} memory pattern $\mathbf{u}^{(k)}$, while the negative branch is essentially an open circuit, and vice versa.

Summation of M terms in the evolution equation is done by current summing. The final results are two currents I_i^+ and I_i^-, which need to be compared by the threshold circuit to determine the sign of the i^{th} bit of the next state pattern x_i'.

5.4.3 Threshold Circuit

The function of the threshold circuit is to generate VDD or GND, depending on whether or not I_i^+ is greater than I_i^-. Thus, any differential amplifier is sufficient. Figure 5.7 depicts the top half of a simple differential amplifier, which can be integrated with the circuit in Figure 5.6 to decide x_i'.

5.4.4 Putting It All Together

In order for easy VLSI implementation, we design a basic ECAM cell that realizes all aforementioned computation. The idea is to draw the correlation computation circuit and the exponentiation, multiplication, and summation circuit, and then extract a basic repeating block. This block, together with a RAM cell, makes up the basic ECAM cell as illustrated in Figure 5.8. The final design of an exponential correlation associative memory that holds M N-bit memory patterns can be obtained by replicating the basic ECAM cell in the horizontal direction M times and in the vertical direction N times.

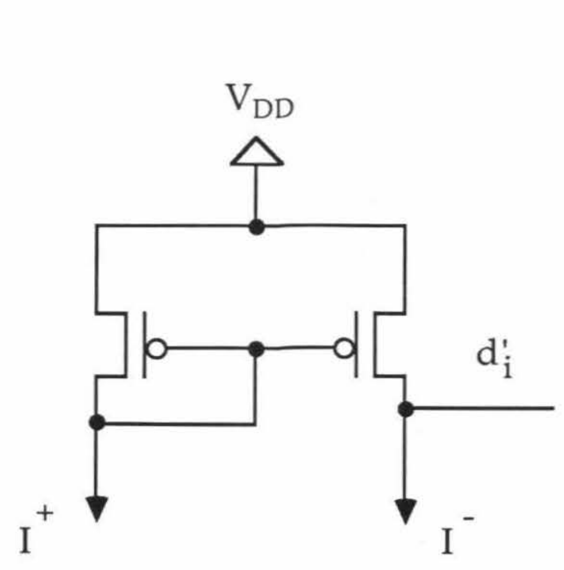


Figure 5.7: Circuit diagram of the thresholding circuit in the ECAM chip

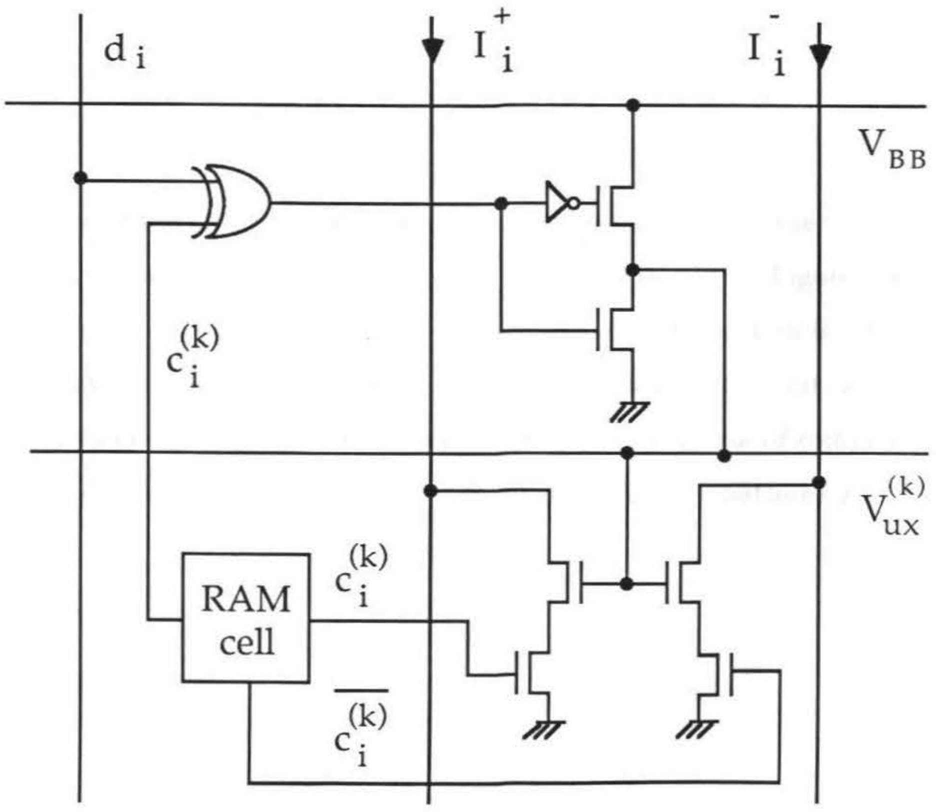


Figure 5.8: Circuit diagram of the basic ECAM cell

5.5 AnaLOG Simulation Results

Before we proceed to lay out the design of ECAM and have it fabricated, some more simulation needs to be conducted. A functional simulator for neural VLSI systems — "AnaLOG" [4] is used to simulate an $M = N = 8$ ECAM programmed with the following memory patterns :

$$\begin{array}{lrrrrrrrr}
\mathbf{u}^{(1)}: & -1 & 1 & -1 & -1 & -1 & 1 & -1 & 1 \\
\mathbf{u}^{(2)}: & 1 & -1 & -1 & 1 & -1 & 1 & -1 & 1 \\
\mathbf{u}^{(3)}: & 1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 \\
\mathbf{u}^{(4)}: & 1 & 1 & 1 & -1 & 1 & 1 & -1 & 1 \\
\mathbf{u}^{(5)}: & 1 & -1 & 1 & 1 & 1 & -1 & 1 & -1 \\
\mathbf{u}^{(6)}: & 1 & 1 & -1 & -1 & 1 & -1 & -1 & 1 \\
\mathbf{u}^{(7)}: & 1 & 1 & -1 & -1 & 1 & 1 & 1 & 1 \\
\mathbf{u}^{(8)}: & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1
\end{array}$$

Because of the memory and execution time constraints, only the first output bit is simulated; i. e., only circuitry that computes x_1' is simulated (see Figure 5.9). The first input pattern is $(-1\ -1\ -1\ \ 1\ -1\ \ 1\ -1\ \ 1)$, the second input pattern is the first with its second bit reversed, and the third input pattern is the same as the first (see how INPUT2 in Figure 5.10 changes from -1 to 1 and to -1 again). For ease of comprehension, the correlations of the three input patterns with all eight memory patterns are listed below.

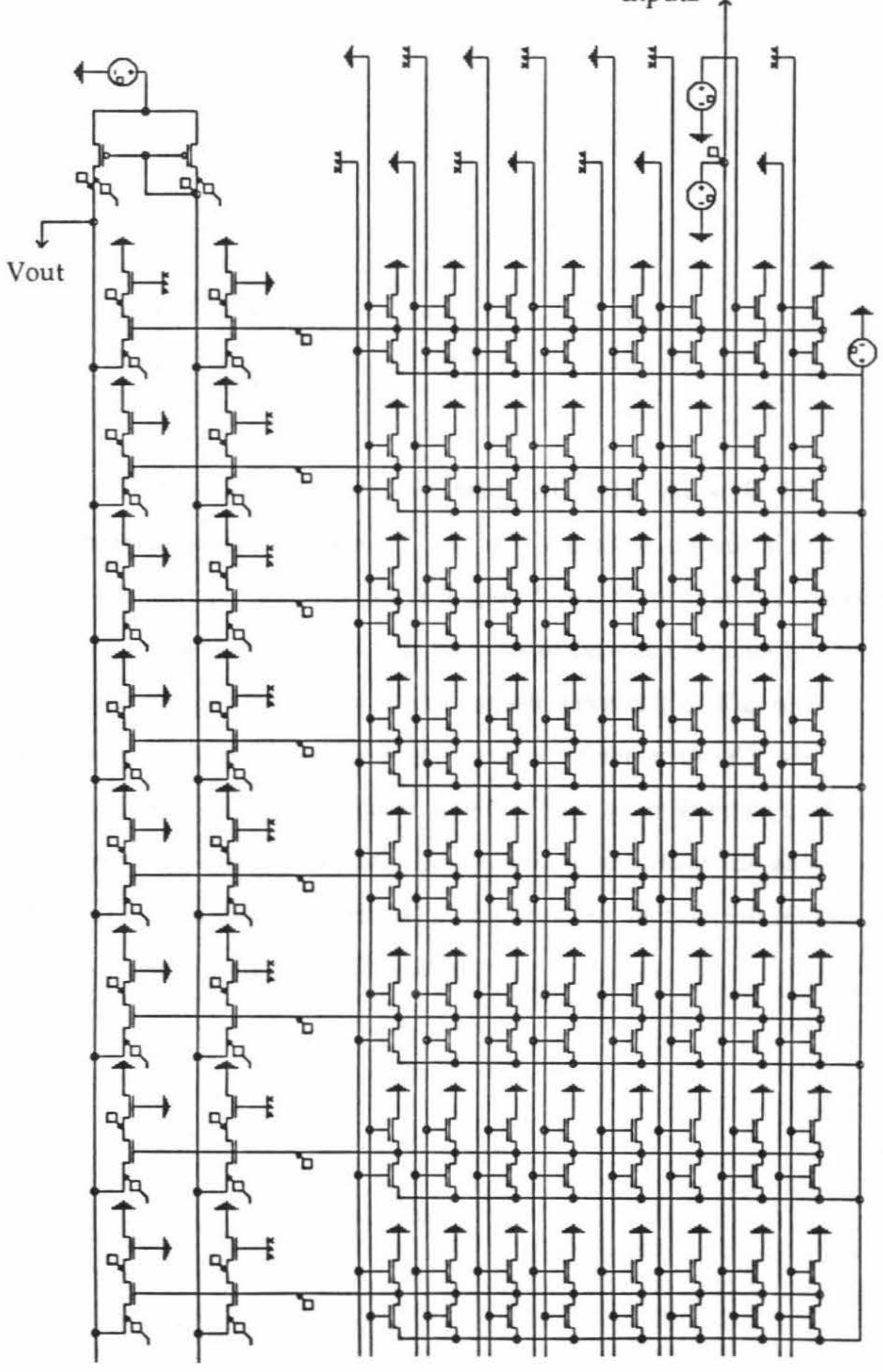


Figure 5.9: Circuit diagram of an $M = N = 8$ ECAM used in AnaLOG simulation

$< \mathbf{u}^{(1)}, \mathbf{x} >$	:	4		6		4
$< \mathbf{u}^{(2)}, \mathbf{x} >$	:	6		4		6
$< \mathbf{u}^{(3)}, \mathbf{x} >$	:	-4		-2		-4
$< \mathbf{u}^{(4)}, \mathbf{x} >$	:	-2		0		-2
$< \mathbf{u}^{(5)}, \mathbf{x} >$	:	-4	$\rightarrow$	-6	$\rightarrow$	-4
$< \mathbf{u}^{(6)}, \mathbf{x} >$	:	-2		0		-2
$< \mathbf{u}^{(7)}, \mathbf{x} >$	:	-2		0		-2
$< \mathbf{u}^{(8)}, \mathbf{x} >$	:	0		-2		0

In Figure 5.10, $+1$ is encoded as 5V, while -1 as 0V. Since the first components of all memory patterns except $\mathbf{u}^{(1)}$ are $+1$, and $\mathbf{u}^{(2)}$ is nearest the initial input pattern, one expects $x'_1 = +1$ initially (VOUT = 5V in Figure 5.10). Note that the correlation-matrix associative memory where $f(t) = t$ will have an erroneous outcome in this case. Next, when the second component of the input pattern (INPUT2 in Figure 5.10) is switched to $+1$, $\mathbf{u}^{(1)}$ becomes the nearest memory pattern. Even with $\mathbf{u}^{(2)}$ also being near, x'_1 still becomes -1 very quickly. Similarly, one sees that when INPUT2 is brought down to 0V, VOUT goes to 5V almost instantly. So, we are assured that the analog computation circuits for associative recall described in the previous section are functionally correct when they are used to build a small ECAM.

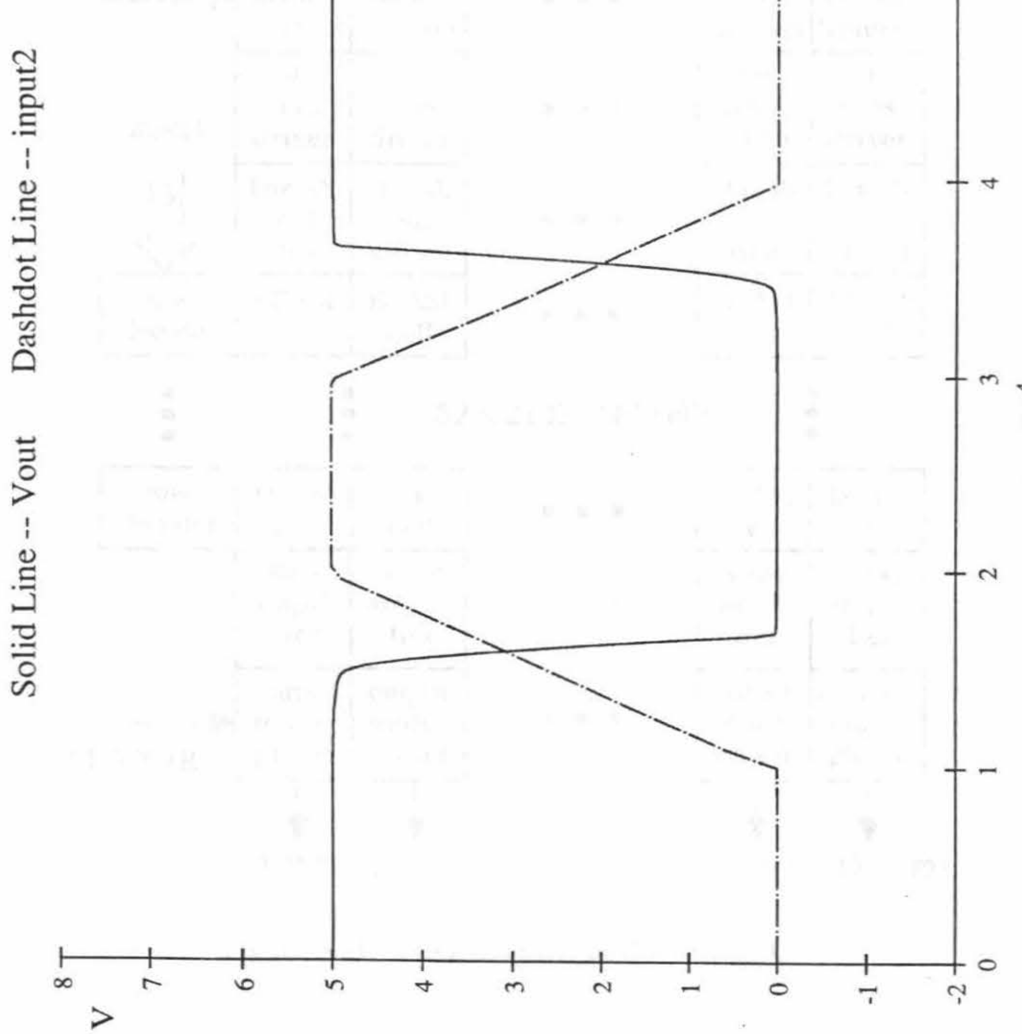


Figure 5.10: AnaLOG simulation results of an $M = N = 8$ ECAM

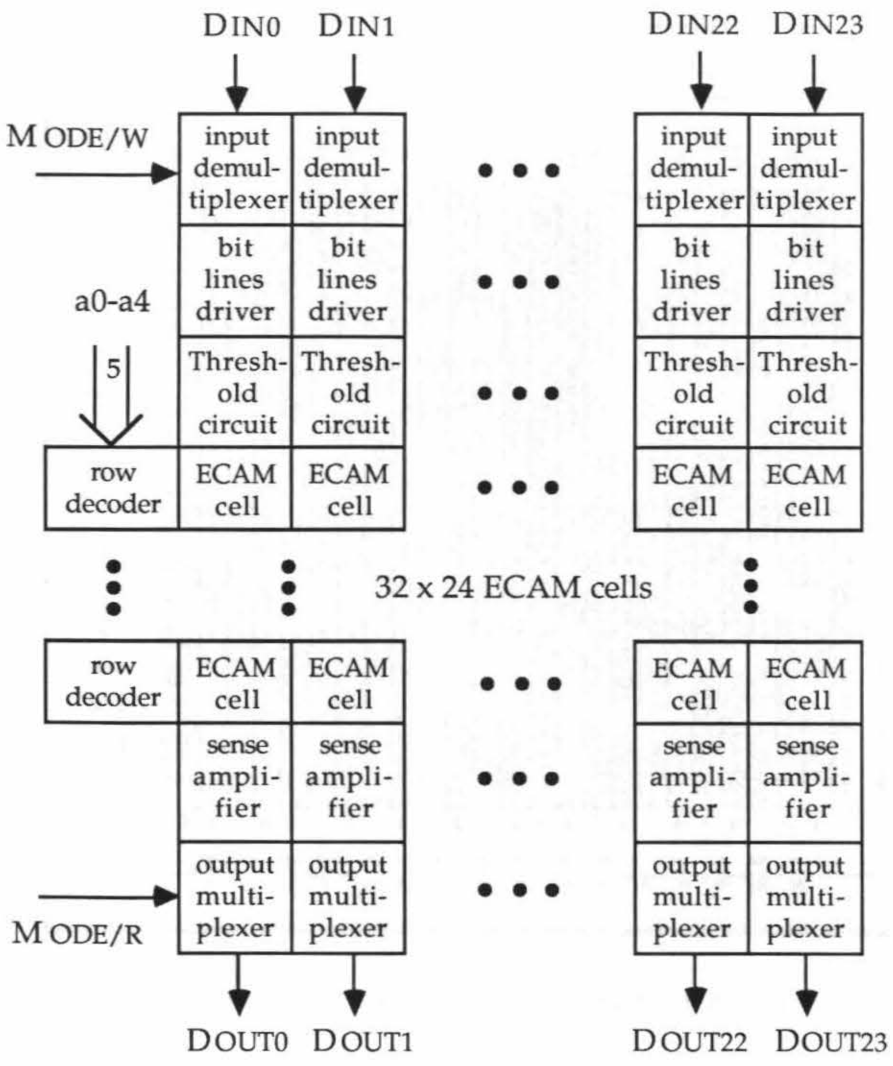


Figure 5.11: Block diagram of the ECAM Chip

5.6 The ECAM Chip and Testing Results

We have explained in detail the circuit design of the proposed ECAM chip; now we illustrate, in Figure 5.11, a block diagram of the complete ECAM chip, including ECAM cells, read/write circuit, sense amplifiers, row decoders, and I/O multiplexers.

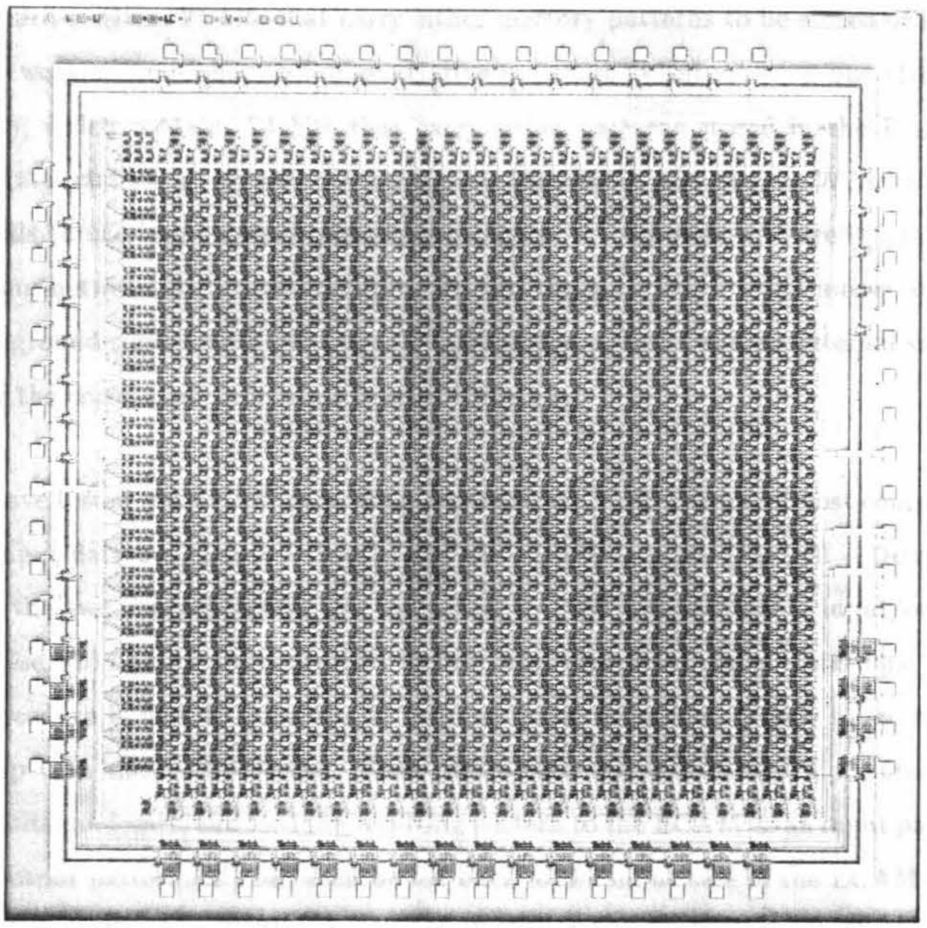


Figure 5.12: Microphotograph of the ECAM chip

We start out by laying out the basic ECAM cell using the MAGIC VLSI design editor. The cell turns out to be 100λ by 140λ. A nonprogrammable (ROM type) ECAM cell is also designed, and it occupies only 32λ by 32λ. Then all other peripheral circuits, such as row decoders, multiplexers, etc. are laid out and simulated by the MOSSIM switch-level simulator. The complete chip is made up of 32×24 ECAM cells, which hold 32 memory patterns each 24 bits wide. It is then sent to MOSIS for fabrication (see the microphotograph in Figure 5.12).

There are four groups of signals on the ECAM chip : 1) Input data bus (DIN0 – DIN23), which contains 24 bits that carry either memory patterns to be stored or input patterns **x** working as probes to the associative memory; 2) output data bus (DOUT0 – DOUT23), which contains 24 bits that carry either patterns stored in the RAM or the next state pattern $\mathbf{x}'$; 3) two control signals (MODE/R and MODE/W, which are select signals to the input demultiplexers and output multiplexers and are responsible for switching between the programming mode and the associative recall mode; and 4) power and ground signals for digital circuits and analog circuits, and an external supply voltage for the correlation computation circuit (VBB).

We have tested the ECAM chip using a specialized VME bus based host computer. Both the input data bus (DIN0 – DIN23) and the output data bus (DOUT0 – DOUT23) as well as the two control signals are connected to 8255 programmable input/output chips. These 8255 chips are controlled by the host computer via a VME bus. The testing procedure is first to generate 32 memory patterns randomly and program the ECAM chip with these 32 patterns. Then pick a memory pattern and flip a specified number of bits randomly, and feed the resulting pattern to the ECAM as an input pattern ($\mathbf{x}$). The output pattern ($\mathbf{x}'$) can then be fed back to the input side of the ECAM chip. This iteration continues until the pattern on the input bus is the same as that on the the output bus, at which time the ECAM chip has reached a stable state. We select 10 sets of 32 memory patterns and for each set we run the ECAM chip on 100 trial input patterns with a fixed number of errors. Altogether, there are 1000 trials tested.

In Figure 5.13, we illustrate the testing results of the ECAM chip. The number of successes is plotted against the number of errors in input patterns for the following four cases : 1) The ECAM chip with VBB = 5V; 2) VBB = 2V; 3) VBB = 1V; and 4) a simulated ECAM with the constant in the exponentiation, a, equals 2. It is apparent from Figure 5.13 that as the number of error increases, the number of successes decreases, which is expected. Also, one notices that the simulated ECAM is by far the best one, which is again not unforeseen because the ECAM chip is, after all, only an approximation of the ECAM model and thus will definitely do more poorly. What is really unexpected is that the best performance is given by the case when $V_{BB} = 2V$ instead of the case when $V_{BB} = 1V$ (V_{TH} in this CMOS process) as we predicted.

This phenomenon is actually the result of two contradicting effects brought about by increasing V_{BB}. On the one hand, increasing V_{BB} increases the dynamic range of the exponentiation transistors in the ECAM chip. Suppose that the correlations of two memory patterns $\mathbf{u}^{(l)}$ and $\mathbf{u}^{(k)}$ with the input pattern $\mathbf{x}$ are t_l and t_k, respectively, where $t_l > t_k$; then

$$V_{ux}^{(l)} = \frac{t_l \, V_{BB}}{N}, \qquad V_{ux}^{(k)} = \frac{t_k \, V_{BB}}{N}.$$

Therefore, as V_{BB} increases, so is the difference between $V_{ux}^{(l)}$ and $V_{ux}^{(k)}$, and $\mathbf{u}^{(l)}$ becomes more dominant than $\mathbf{u}^{(k)}$ in the weighted sum of the evolution equation. Hence, as V_{BB} increases, the error correcting ability of the ECAM chip should improve. On the other hand, as V_{BB} increases beyond the threshold voltage, the exponentiation transistors leave the subthreshold region and might enter saturation, where the drain current is approximately proportional to the square of the gate-to-source voltage. Since a second-order correlation associative memory in general possesses a smaller storage capacity than an ECAM, one would expect that with a fixed number of loaded memory patterns, ECAM should have a better error correcting power than the second-order correlation associative memory does. In conclusion, two contradicting effects are going on as V_{BB} is raised; one tends to enhance the performance of the ECAM chip, while the other tends to degrade it. A compromise of these two effects is reached, and the best performance is rendered when $V_{BB} = 2V$ as shown in Figure 5.13.

In the case when $V_{BB} = 2V$, the drain current versus gate-to-source voltage characteristic of the exponentiation transistors is actually a hybrid of a square function and an exponentiation function : At the bottom it is of an exponential form, and it gradually flattens out to a square function, once the gate-to-source voltage becomes larger than the threshold voltage. Therefore, the ECAM chip with $V_{BB} = 2V$ is a mixture of the second-order correlation associative memory and the pure ECAM. According to

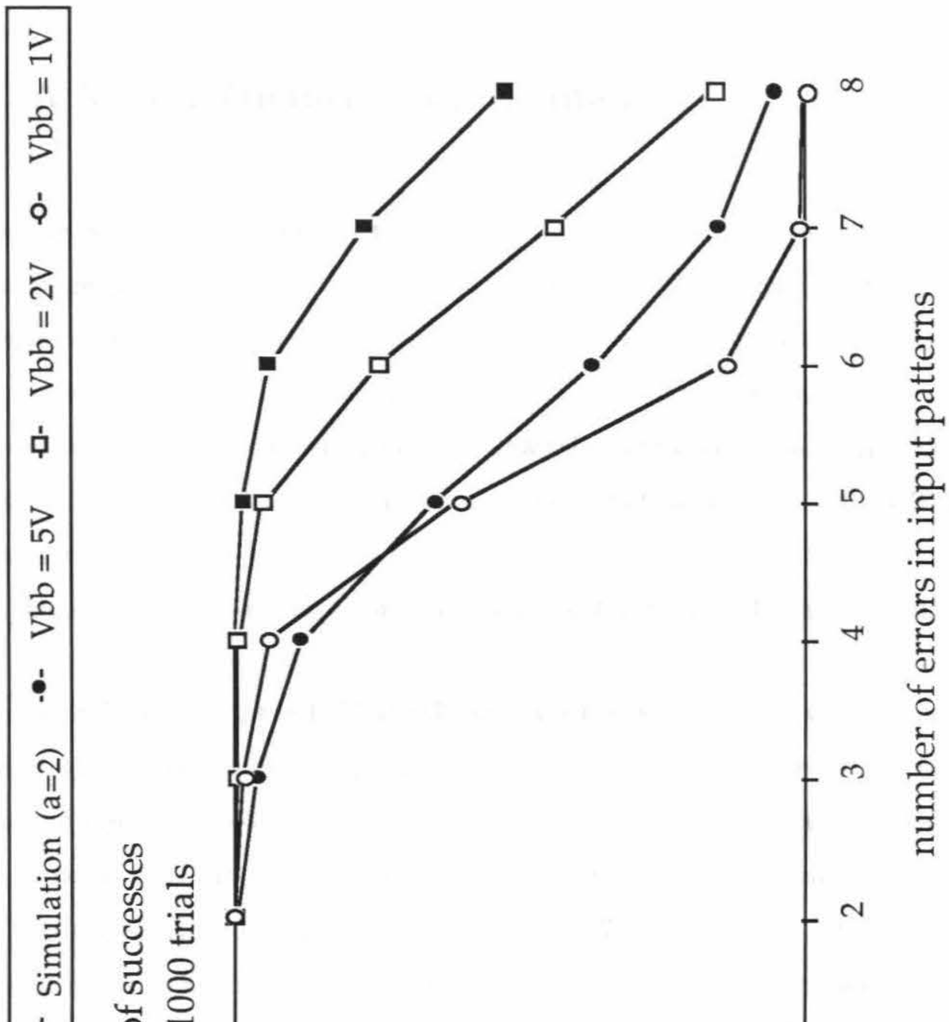


Figure 5.13: Comparison of the ability to correct errors of the ECAM chip with different VBB and a simulated ECAM with $a = 2$

the convergence theorem for correlation associative memories and the fact that f in the ECAM chip with $V_{BB} = 2V$ is still monotonically nondecreasing, the ECAM chip is still asymptotically stable in the synchronous updating mode even when $V_{BB} = 2V$.

5.7 A Vector Quantization Example

In order to measure the speed of the ECAM chip for real applications, we choose the binary image vector quantization as an example problem. Vector quantization is a means of data compression (source coding) on information to be transmitted or stored, e. g., speech waveforms, images, etc. [3]. In principle, a vector quantizer should, given a set of codewords and an input, find the nearest codeword to the input. Then only the index of the nearest codeword is transmitted or stored instead of the information itself. Usually, the number of possible codewords is much less than that of possible information patterns; hence, vector quantization can reduce the bandwidth (number of bits) needed.

The problem which the ECAM chip solves is the binary image vector quantization problem, where pixels in the images are either black or white. At first input images are partitioned into 4×4 blocks, and each block is vector-quantized by the ECAM chip. A set of 32 codewords are chosen, and they correspond to all white, all black, horizontal edge, vertical edge, and diagonal edge blocks as shown in Figure 5.14. The ECAM chip is programmed with these codewords, and 4×4 blocks from a binary image are fed to the ECAM chip. A reconstructed image is formed by replacing each block by its corresponding codeword. However, there are times when the output pattern of the ECAM chip is not a codeword (remember — there are many spurious stable states in any associative memory), in which case an all white block is generated instead. Figure 5.15 and Figure 5.16 illustrate two original binary images and their ECAM chip reconstructed images. It is obvious that the quality of the reconstructed binary images is not as good as the originals, yet this is the price paid for reduced transmission or storage bandwidth.

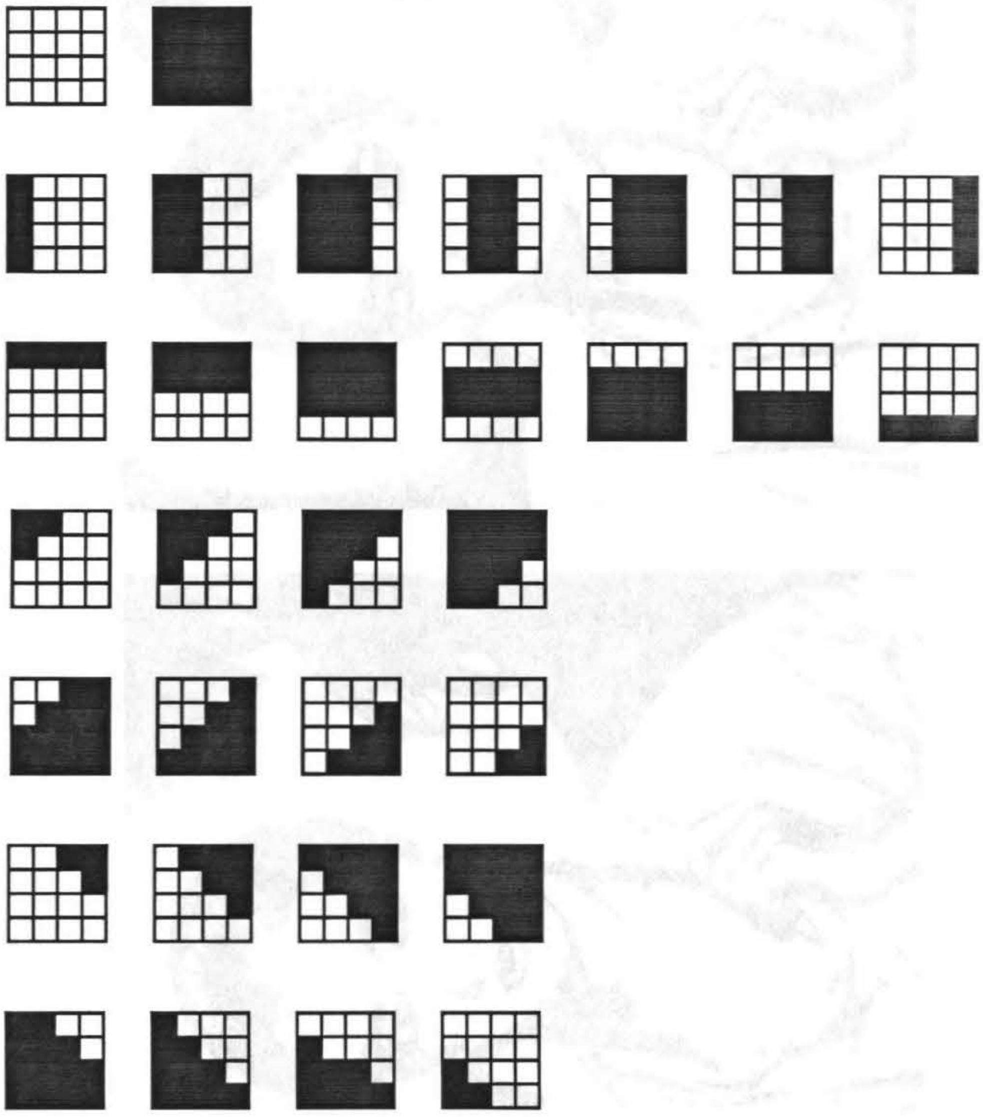
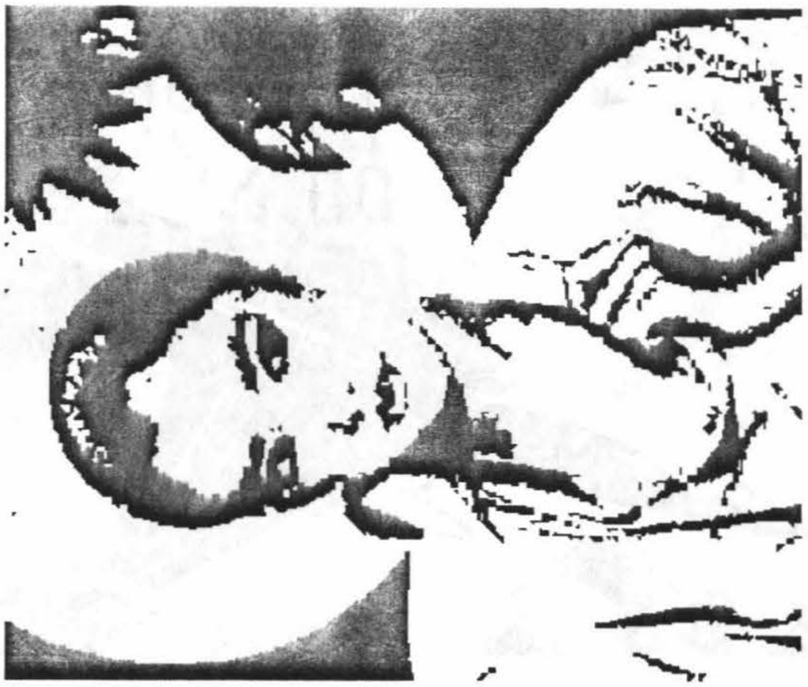


Figure 5.14: 32 codewords used in binary image vector quantization

(a)



(b)

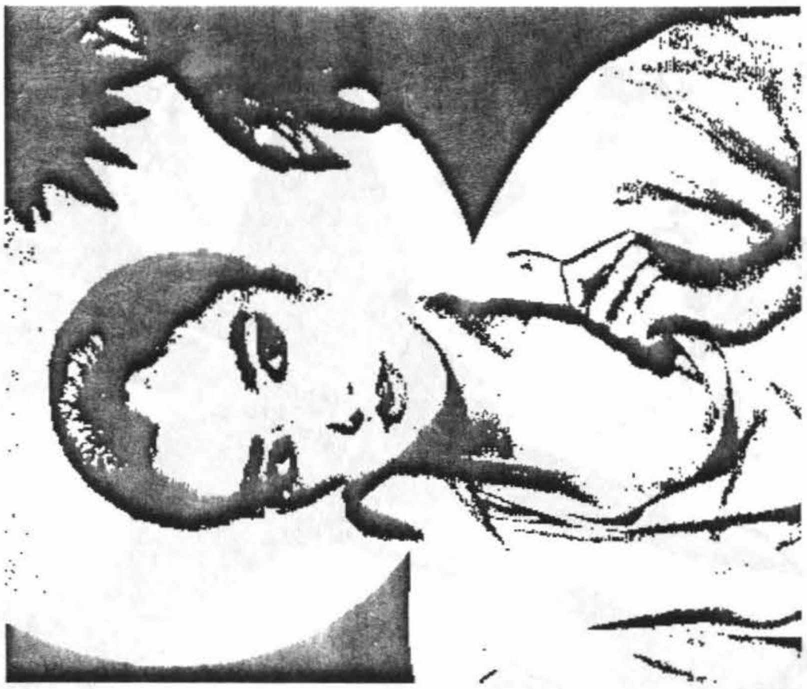
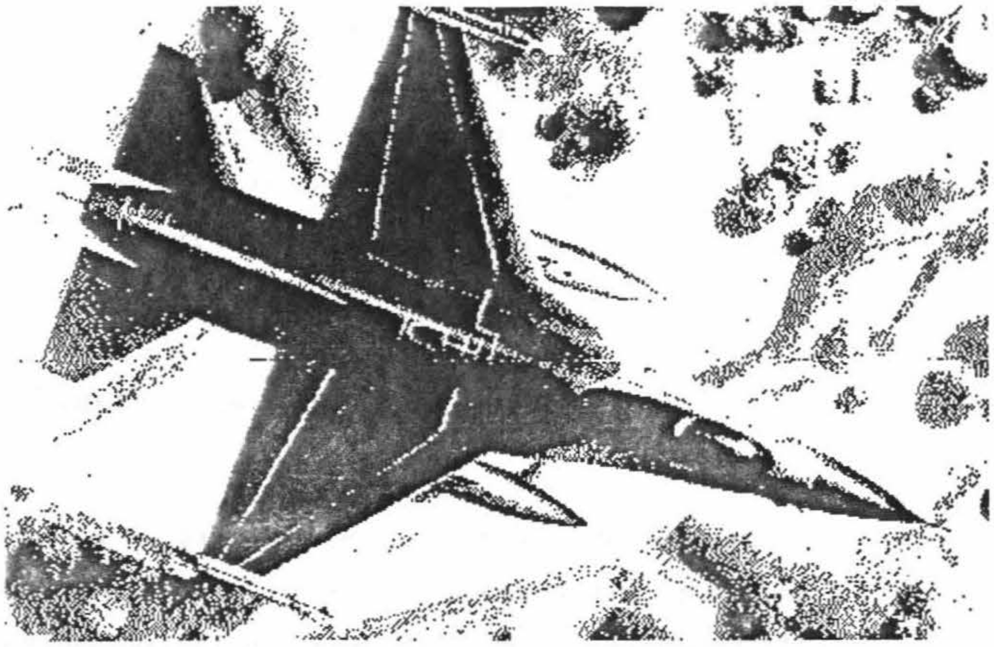


Figure 5.15: Comparison of (a) the original girl image and (b) the reconstructed girl image after vector quantization by the ECAM chip

(a)



(b)

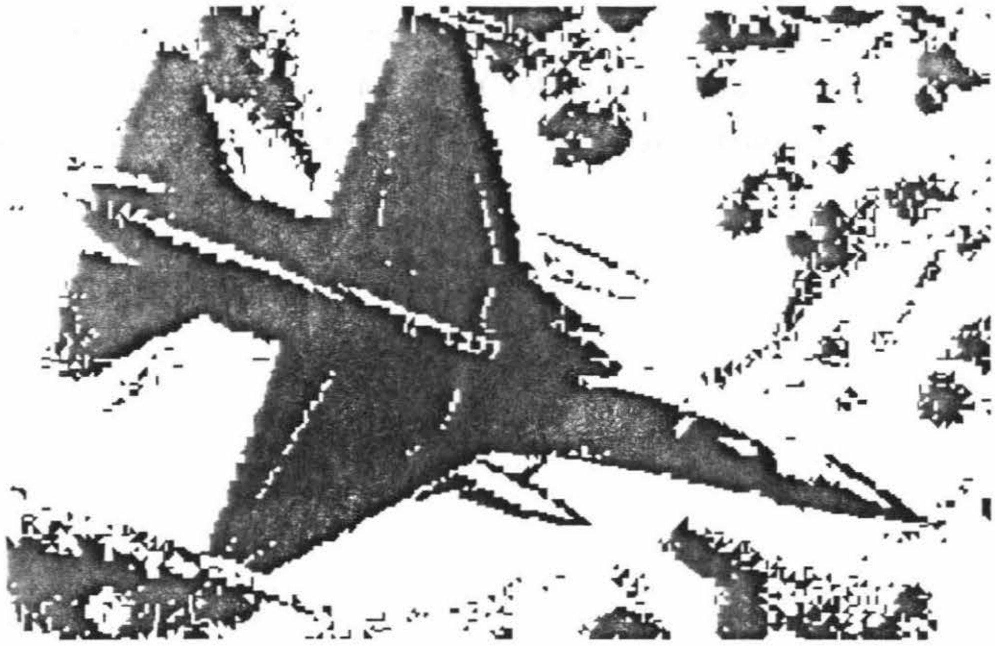


Figure 5.16: Comparison of (a) the original airplane image and (b) the reconstructed airplane image after vector quantization by the ECAM chip

One thing worth noting is the speed that the ECAM chip can vector-quantize these binary images. We find that the ECAM chip is capable of doing one associative recall operation in less than 3 μs (this includes the overhead for the ECAM chip to communicate with 8255 chips). This projects to about 28 ms for the 416×352 pixel girl image in Figure 5.15, or more than 30 images per second. For larger images, more ECAM chips can work together since each block is quantized independently.

5.8 Conclusions

In this chapter, we have presented a circuit design for implementing the exponential correlation associative memory proposed in Chapter four. In addition, a VLSI chip for this design is fabricated and tested. The performance of the ECAM chip is shown to be almost as good as a simulated ECAM. Finally, the speed of the chip is measured by employing it to do vector quantization on binary images. And it is found that the ECAM chip can process binary images in real time, i. e., faster than 20 – 30 images every second. In conclusion, we believe that the ECAM chip provides a fast and efficient way for solving associative recall problems and minimum distance classification problems.

References

[1] T. D. Chiueh and R. M. Goodman, "High-Capacity Exponential Associative Memory," in *Proc. of IEEE ICNN*, Vol. I, 1988, pp. 153–160.

[2] L. A. Glasser and D. W. Dopperpuhl, *The Design and Analysis of VLSI Circuits.* Reading : MA, Addison-Wesley, 1985.

[3] R. M. Gray, "Vector Quantization," *IEEE ASSP Magazine*, Vol. 1, 4–29, 1984.

[4] J. P. Lazzaro, "AnaLOG : A Functional Simulator for VLSI Neural Systems," Technical Report, 5229:TR:86, Computer Science Department, California Institute of Technology.

[5] C. A. Mead, *Analog VLSI and Neural Systems.* Reading, MA : Addison-Wesley, 1989.

[6] O. Minato, T. Masuhara, T. Sasaki, H. Nakamura, Y. Sakai, T. Yasui, K. Vehihori, "2K x 8 Bit Hi-CMOS Static RAMs," *IEEE Journal of Solid-State Circuits*, Vol. SC-15, 656–660, 1980

[7] N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design, A System Perspective.* Reading, MA : Addison-Wesley, 1985.