A STUDY ON ASSOCIATIVE MEMORY CLASSIFIER AND ITS APPLICATION IN CHARACTER RECOGNITION

Ming Zhang

A Thesis
in
The Department
of
Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy at
Concordia University
Montreal, Quebec, Canada

October 1992

ISBN 0-315-84671-2

ABSTRACT

A STUDY ON ASSOCIATIVE MEMORY CLASSIFIER AND ITS APPLICATION IN CHARACTER RECOGNITION

Ming Zhang, Ph. D.

Concordia University, 1992

A novel neural network classifier, called associative memory classifier, is developed in this thesis modeled upon associative memory network. It is studied for its application potential in the recognition problem of large number categories, such as that of Chinese characters.

The major findings of this work are in three aspects. First of all, it is found that a feed forward associative memory network can become a suitable pattern classifier by an appropriate selection of its output vectors, called inner codes. The classification ability of an associative memory classifier is determined ultimately by the distinctiveness of its input patterns, but a set of properly selected inner codes may help the classifier to approach the limit of its capability.

In the next place, seeking for better inner coding schemes should be in connection with each specific case on the basis of input patterns' characteristics. Attempts

to resolve this problem by enumeration will lead to a non-polynomial complexity which is computationally infeasible. On the other hand, real life data are usually mathematically undescribable. Hence, a practical way to find such schemes is to work out an optimization strategy first under some ideal conditions, then apply it to the real data with remedial measures.

Thirdly, when an associative memory network is used as a pattern classifier, if the feature patterns are transformed into a form more suitable to it, the performance of the entire system can be improved significantly. Also, due to the parallel computation mode of neural networks, data reduction is not a problem any longer. Therefore in addition to stable feature detection, the other objective in feature extraction here has been changed to the improvement of feature patterns suitability to the currently used neural classifier.

All these findings are verified by computer simulation with sets of common and similar multi-font Chinese characters. Our experiments are conducted presently using more than two hundred and fifty categories for a single-level classifier. To test the robustness of the system and make it meet the needs of practical use, similarities are introduced into the testing data by characters which look alike and printed in different fonts.

To All Those That Deserve

ACKNOWLEDGMENTS

I would like to express my appreciation to Professors C. Y. Suen and T. D. Bui, my supervisors at Concordia University, for their advice and counsel during the preparation of this thesis.

I wish to thank Dr. L. Xu for his useful discussions and suggestions during his visit at Concordia University. I am also thankful to Mr. W. Wong, Mr. C. Yu and all other staff members in the Departments of Computer Science and Computer Services of Concordia University for providing me with the simulation environment in my pursuit of this research. I am deeply grateful to my parents for their support and encouragement, especially the instruction of my father in scientific research.

I greatly appreciate Professor Y. Wei, my ex-supervisor and the President of Southeast University, for her dedication to the Joint Doctoral Program between Concordia University and Southeast University of Nanjing, China, under which this work was conducted. I also wish to display my gratitude to all those people from the two universities who are involved in the joint program.

Finally, I would like to acknowledge the financial support received from the Canadian International Development Agency, the Natural Sciences and Engineering Research Council of Canada, the Ministry of Education of Quebec, and the Centre de Recherche Informatique de Montréal.

Contents

List of Figures

List of Tables

Chapter 1

Introduction–Chinese Character Recognition and Neural Networks in Character Recognition

Human brain, the biological model of artificial neural networks (briefly neural networks), manifests an excellent performance in pattern recognition, especially for the problems involving a large number of classes, *e.g.* characters of Chinese and some other oriental languages. This has generated a great interest in the application of neural networks to pattern classification. This thesis explores a neural network classifier suitable for solving the recognition problem involving a large number categories, with its application in Chinese character recognition. In this connection, it is indispensable, first of all, to give a brief summary about both the problem and the major steps of Chinese character recognition, as well as to review the general situation on the application of neural networks to character recognition.

1.1 The Chinese Character Recognition Problem

Chinese character recognition is a very important area in the field of pattern recognition. The large number of Chinese characters, the great variety of their shapes and the complexity of their structures, as well as the diversity of ways in which they can be differentiated from one another, make it a typical recognition problem involving a large number of categories, and therefore very attractive and challenging. Computer recognition of Chinese characters is considered to be a very hard problem and regarded as one of the ultimate goals of character recognition research [1]. In addition, the facts that they have thousands of years of history and are used by about one fourth of the world's population today, suggest that their automatic recognition has a great potential in commercial application.

The early studies of Chinese character recognition dated back to the 1960's [2]. Since then, a specific research topic in pattern recognition has emerged gradually with the appearance of hundreds of papers written on it and also some practical systems in the market. All these could be traced through a series of survey articles and books published during the last three decades [3]–[10]. A more recent and comprehensive review of the advancement of the different aspects of Chinese language computing is provided by [11] in which Chinese character recognition is considered as an efficient computer input technology. Besides, to enlarge the scope from Chinese character recognition to the entire field of character recognition, a global overview is provided in [1], for it does not only review the research work in the recognition of different character sets which include a specific section dedicated to that of Chinese characters.

but also covers extensively the topics from methodologies in character recognition, to the application of character recognition techonology, and the practical optical character recognition (OCR) system.

In recent years, another area of information science, namely, artificial neural networks, has gained an astonishing interest of many researchers. Not only many new models of neural networks, but also their application in many different areas including pattern recognition, are being studied [12]. Numerous investigations have already been reported on the recognition of numerals and letters using neural networks, and inspiringly, that of Chinese characters has started as well.

1.2 Major Phases in Chinese Character Recognition

Generally speaking, the process of Chinese character recognition is a combination of techniques used in image processing and pattern recognition. Some preprocessing steps should be done to prepare the character data for recognition. These include thresholding which is needed to obtain the binary representation of the patterns, some measures to remove noise in the original character images, skeletonization, size normalization, *etc.*, to facilitate the extraction of features. However, from the recognition point of view, the process of Chinese character recognition has merely two major phases, namely, feature extraction and character classification.

1.2.1 Feature Extraction

Feature extraction plays an important role in Chinese character recognition since it is one of the decisive factors in the success of the whole process. According to the way in which they are extracted, features can be divided into two types, *i.e.* global and local ones [9].

Global features are usually obtained from some other new domains into which various mathematical transformations map the character images. Although their principle is rather simple, global feature approaches need large storage and long computational time especially under the sequential computation mode. Typically, some orthogonal transformations such as Fourier [13], Walsh [14], and Hotelling [15], [16] transforms have been chosen for this purpose.

Local features are selected based on the geometrical and topological properties of the characters. Except for their sensitivity to noise, these approaches are powerful techniques especially for handwritten Chinese characters. Some of the most often used local feature extraction methods are: stroke distribution [17], [18], stroke analysis [19] [21], peripheral feature detection [22], background feature and feature point distribution [23], [24], *etc.*

There also exist some other feature extraction methods, for instance, the Hough transform technique [25], with which a transformation of character matrix is made from the spatial domain into the parametric domain to detect strokes; it combines the advantages of both global and local feature approaches, making it not only insensitive

to the variance of character images, but also not time consuming and complex in computation.

1.2.2 Pattern Classification

For efficiency, a recognition system for a large variety of patterns such as that for Chinese or some other oriental characters is usually organized as a hierarchical classifier. This could be done in one of two ways presently. The first is a two stage approach which is designed to reduce the size of the candidate set for classification in the second stage [2]. The rest is a multi-level classifier such as a decision tree which uses a different subset of features in each of its layers [26], [27]. Incidentally, a comprehensive discussion on decision tree classifier is made in [28], which also institutes a comparison between decision trees and neural networks.

In the hierarchical classifiers, different kinds of features are often used in the different layers. This occurs most probably in the former case. Another aspect which is worth mentioning is the way the hierarchical classifiers are organized. As indicated in [7], there are two ways to do this. First, the hierarchy is feature-dependent, that is to say, a series of feature subsets are predefined. They are used sequentially to reduce the candidate set further and further, until finally one candidate emerges. The second way which appeals more to our intuitive logic has a "superclass"-dependent hierarchy. Candidate groupings or "superclasses" are predefined and stored in a structure that could be viewed as a tree. A subset of the features is used to determine the branch of the tree followed at any levels. Eventually the procedure branches to a leaf that

contains only a single character class. This tree-growing algorithm, however, must be devised to determine the features to be used and to define the candidate classes at each stage of the hierarchy.

1.3 Neural Networks in the Recognition of Numerals and Roman Letters

Neural networks offer several advantages over traditional recognition techniques, most notably are their exceedingly fast computation ability due to massive parallelism and great degree of tolerance to distortions provided by learning mechanism. These arouse an interest in the study of their application in the field of pattern classification, and many such investigations have been made in the recognition of numerals and Roman letters, especially in the last five years.

1.3.1 Selection of the Neural Network Model

The potential of many different neural network models in the application of pattern recognition was explored soon after neural network became a hot topic in research [12], [29], [30], including single layer perceptron, associative memory, Hopfield net, multi-layer perceptron, Adaptive Resonance Theory (ART), self-organizing feature map, Learning Vector Quantizer (LVQ), Boltzmann machine *etc.* There are also some neural network models, such as Fukushima's neocognitron, which have been established solely for this purpose.

Some comparison work on the performances among several neural network paradigms has been conducted which may help in the selection of appropriate models for a specific application. One early work of this sort applied to handwritten numeral recognition is made in [31]. The models involved are linear autoassociative system, threshold logic network, back propagation model, Hopfield network and Boltzmann machine. The first three networks are feed forward classifiers which more or less resemble to a classical perceptron while the last two define a class of content addressable memory systems. The order of these models represents a graduated progression of their complexity, and their performance declines with it. This is because, claimed by the authors, when a model increases in complexity, it becomes much more difficult to control and also impossible to be tested exhaustively under all possible configurations and parameter settings.

In the meantime, Bisset *et al.* conduct another comparative study on the recognition behaviour of different neural networks about a set of machine-printed numerals [32]. The models under consideration include probabilistic logic node, back propagation model, and ART network. The experimental results indicate that the back propagation model provides a much better recognition performance than the other two models. Also, the authors have proposed some standards for the comparative work of this sort to be carried out, such as: recognition rate, processing speed, the number of training cycles and, the number of training patterns needed to obtain an acceptable level of recognition performance. It is argued that this kind of comparative study undertaken with respect to a real-world problem has the value of focusing

attention on the sort of design factors which are likely to be of primary relevance in the implementation of a practical system.

1.3.2 Back Propagation Model

Precisely, the so-called back propagation model should be named as multi-layer perceptron trained by back propagation learning algorithm. In the application of neural networks, the back propagation model is the one most commonly used in character recognition [33]-[55]. Originally proposed by Rumelhart *et al.* [56], it is actually a powerful learning algorithm which can deal with multi-layer neural network problems. The importance of the back propagation model is shown by Lippmann [29] who argues that such a network with merely two hidden layers is sufficient to form arbitrarily complex decision regions and therefore to separate populations of patterns even though their distributions might be intermeshed spatially in pattern space.

Application in the Recognition of Numerals and Letters of the English Alphabet

The recognition of handwritten numerals has already been studied extensively under different neural network models, especially that of back propagation. It is chosen as one of the benchmarks for the research in the application of neural networks to realistic problems for its clearly defined commercial importance and the level of difficulty that makes it challenging, and yet not so large as to be completely intractable [33].

As an application oriented example, handwritten zipcode recognition by charac-

ter classification systems based on back propagation network is discussed in [31] [41] Krzyzak *et al.* [31], [35] propose a modified back propagation learning algorithm by changing its error function. The non-convergence problem in the original back propagation algorithm is eliminated and the process of training is two times faster. The method of feature extraction in this system, however, still remains in its traditional way with the use of Fourier descriptors and other topological features.

On the other hand, the authors of [37]-[41] demonstrate in the development of their work the ability of back propagation networks to deal with a large amount of low-level representation of data through the construction of a network that can be fed with images directly without an explicit feature extraction stage. This capability is achieved through a bi-pyramidal architecture of the network and the constraints on the weights designed with the consideration of some geometric knowledge about shape recognition. In fact, here the function of local shape like feature detection has been incorporated into the network. Meanwhile, some hardware requirements for the implementation of such neural network systems are discussed, a neural net chip designed to its help is described [42]-[44], and the effort on this continues [57]. Finally, in both of the above two systems, recognition rates of about 90% have been reached.

A similar study has been carried out by Yamada and his colleagues on a set of handwritten numerals [45]. In order to solve the same problem addressed in [34] and [35], they propose another scheme to improve back propagation algorithm, that is, to evade the standstill in learning by controlling the slant parameter of the sigmoid

function. Three neural networks are used in their test. They are globally connected network and locally connected network with grey character image input and a network which classifies characters based on their contour features. Experiments show that the improved algorithm can raise the learning speed by three times, and the recognition rates of 98.3%, 98.8%, and 99.1% have been obtained respectively. Besides, it is also found that the locally connected network can extract global features more flexibly than a globally connected network, and in order to self-organize the feature extraction process, it is necessary to construct an innate structure, like local connections, into a neural network.

Apart from those networks described above, Burr has also conducted a number of experiments to assess the performance of back propagation networks in speech and handwriting recognition of digits and alphabetic characters [46], [47]. Nearly the same level of accuracy has been achieved in his neural systems with nearest neighbour classifiers. This result is also supported by the work of Weideman *et al.* [48]. They conclude from their experiments that except its computational attractiveness, the back propagation model is analogous to the nearest neighbour classifier. However, for some complex and difficult problems, it is indeed more efficient in both storage and operation requirements than some conventional classifiers. Another such comparison of classical and neural classification methods is made in [49]. Here, both the polynomial classifier and the multi-layer perceptron with back propagation learning algorithm are under discussion. In addition to almost the same results as has just been presented, the similarity and difference in these two approaches, particularly

their mathematical essence, are specified in detail.

The back propagation model is also the most commonly used neural network type in constructing classifiers for the recognition of the English alphabet [50]-[54]. The system proposed in [50] has no explicit feature extraction phases. Instead, it has one more step in preprocessing, called coarse coding, which is believed to be analogous to optical blurring, corresponding to certain neurophysiological mechanisms such as the retinotopic maps present in the early stages of the human visual system. This may lead to an increase in robustness of the system and its extensibility to the recognition of any form of character image without the need for explicit awareness of the features distinguishing them. On the other hand, traditional feature extraction methods have been implemented in both [51] and [52]. Six features derived from the geometrical moments of the image are used in the former which are invariant to translation, rotation and scaling. The latter, however, utilizes the Walsh coefficients of the intensity distribution function of the character image. The neural system constructed thereby is free from slight variations in character orientation and has been tested with multi-font alphabets. The recognition rates of 98% or above are reported in both of these two classifiers. Feature extraction is carried out in [53] by an invariance network which is insensitive to translation and rotation. The basic idea is to organize into the network through layers of neurons all forms of possible translation according to image resolution and those of rotation under present consideration. Obviously, this scheme is too costly since a tremendous amount of neurons is needed in such a network for practical use. Last of all, in [54], a quantized learning rule with unipolar

binary weights which is useful in a three-layered opto-electronic neural hardware is reported, and the recognition of twenty six letters of the English alphabet by this neural processor has also been demonstrated.

On the Construction of Back Propagation Classifier

In addition to the above mentioned experiments themselves, some aspects about the structure or the learning process of back propagation model are also addressed which are of theoretical significance.

Perhaps the most important conclusion comes from Lippmann's analysis which demonstrates that no more than three layers (except the input layer) are required in a multi-layer feed forward network because this has already been able to generate arbitrarily complex decision regions [29]. Also, a good insight into the determination of the number of neurons used in each of the two hidden layers is given which shows that they are all related to the complexity of the decision regions of the problem.

Burr offers two other observations in his experiment which are worth mentioning for extending our knowledge about the effect of a back propagation network's structure on its learning process and the recognition performance [47]. First of all, it is found that the number of learning iterations necessary for convergence decreases with an increase of hidden units since additional neurons provide extra degrees of freedom in the form of decision planes. Secondly, the recognition rate increases with the number of hidden units up to a point, no further improvement occurs and even a slight decrease in accuracy can be observed above this point. This is due to the

excessive representational power provided by extra hidden neurons which can model artifacts of the limited statistical sample and reduce the accuracy. A similar result has also been produced by Khotanzad *et al.* [51]. Moreover, the relationship among the number of hidden nodes, the number of separable regions, and the dimensionality of input spaces is elaborated theoretically in [58], which can be utilized as a guideline in the structural design of a feed forward network.

Le Cun and his colleagues have considered in their work the influence of the network's architecture on its generalization ability [37]-[41]. They believe that a good generalization can only be obtained by designing a network architecture that contains a certain amount of *a priori* knowledge about the problem, and a fully connected network with enough discriminative power for the task would have far too many parameters to be able to generalize correctly. In their network, based on some geometric knowledge of handwritten numerals, the principles of restricted connection for local feature detection and weight sharing for the tolerance of slight displacement are introduced. It is also pointed out that the learning time of such a network is relatively short because of these constraints imposed in addition to the redundant nature of the data in training set.

The time-consuming nature is one of its two built-in defects of the back propagation model. The other one is the probable convergence of the training process to local minima rather than the desired global ones. A few general measures have been mentioned to solve this issue [29], such as allowing extra hidden units, lowering the gain term used to adapt weights, and making many training runs starting with differ

ent sets of random weights. Moreover, in dealing with their problems at hand, both groups of Krzyzak and Yamada have proposed their specific ways to settle the local minimum issue. What is interesting is that, in both cases, an increase in learning speed is reported as well.

The learning speed issue is also addressed in [59], which is worth mentioning in spite of the fact that the network is designed to recognize only a few similar Chinese (sometimes called Kanji in Japanese) characters. Mori *et al.* present in this work a new concept of learning environment. They argue that no learning procedure, like that of human beings, exists in isolation in the real world. The learning performance of a network, *e.g.* its speed *etc.*, is highly influenced by some environmental factors such as the order, number, and repetition of training samples. Two rules are proposed to deal with these factors in a learning procedure. The first rule requires the checking of recognition performance during learning, and focus the training on categories that are not well recognized. Then, on a more fine-grained level, the error for each sample is measured, and the greater this error is, the more often that sample is presented. But, to prevent over-training on only a small portion of the samples, the second rule is introduced. It will increase the number of training samples when it is observed that the network's total error rate falls below a certain value. They believe that these rules will lead to more balanced sharing of the network's resources in recognition over categories, and therefore in the improvement of the efficiency of its learning process.

Application in Korean Character Recognition

The back propagation model has also been used for the recognition of printed Korean characters [55]. Due to the large number of patterns in this character set, the recognition system is hierarchically organized with two sorts of functional oriented networks, each of which is a multi-layer perceptron. At the outset, all the characters to be recognized are subject to a type classification network which divides them roughly into six types according to their overall structure. Then, the corresponding one of the six recognition networks will classify them further into character codes in line with the contents of all their different parts. The recognition rate is above 98% for 990 most frequently used Korean characters when the noise included learning is applied. However, although formed roughly in square like Chinese characters, the Korean characters are recognized here based on their phonetic nature. A Korean character is formed by combining two to four symbols drawn from 14 consonants and 10 vowels. Structurally it can be divided into one of the six groups according to the shape of the vowel included and the existence of some consonants. This forms the basis of the architectural design of the whole system. Therefore, the classification of Korean characters here is quite similar to the identification of English words based on the recognition first of each of its componental letters.

1.3.3 Neocognitron

Neocognitron is a neural network model invented by Fukushima solely for the visual pattern recognition [60] [63]. It is a hierarchical network consisting of several

layers of neurons, and capable of recognizing deformation invariant visual patterns. Neocognitron has several advantages:

1. It has a large power of generalization, presentation of only a few typical examples of deformed patterns (or features), rather than all of them which might appear in the future, is enough for the learning;

2. After learning, it can recognize input patterns robustly, with little effect from deformation, changes in size, or shifts in position; and therefore,

3. In contrast to most conventional pattern recognition systems, it does not require any preprocessing.

Application in Alphanumeric Character Recognition

Fukushima *et al.* have used neocognitron in handwritten alphanumeric character recognition [64], [65]. The network has four stages, each of which has one layer of feature extracting cells and a layer of cells to allow for position errors. These layers of cells are divided further into many cell planes, the cells among each of which come to extract one and the same feature at different locations. This system is trained by supervised learning. Typical samples of deformed patterns should be chosen carefully, and the training is performed step by step from the lowest stage to the highest one. The network is designed to recognize 35 alphanumeric patterns, namely, 26 uppercase alphabetic characters and 10 Arabic numerals, where the alphabetic character O and the numeral 0 are treated as the same pattern. The results of computer simulation

show that the neocognitron is able to recognize the input pattern correctly without being affected by deformation, change in size, shifts in position, or contamination by noise.

The implementation of neocognitron on a parallel computer has also been discussed [66], [67]. Under a full consideration of the structure of neocognitron network, the loosely coupled parallel computer with hypercubic connection is selected. A recognition system for handwritten numerals by the neocognitron [62] is realized on a parallel computer, called NCUBE, and the recognition speed of eight characters per second is reached.

Furthermore, Fukushima proposes a selective attention model. The part of this network which fulfils the recognition function is in fact a neocognitron, and its ability has been extended aiming at the recognition and segmentation of connected characters in cursive handwriting [68], [69]. When a composite stimulus consisting of two patterns or more is presented, the model focuses its attention selectively to one of them, segments it from the rest, and recognizes it. After that, the network switches its attention to recognize another pattern. The model can also restore imperfect patterns. A successful performance on the recognition and segmentation of characters has been observed in computer simulation, although the data in the input string are different in shape from the training samples. This is of considerable significance practically, for even the same character is written differently when it appears in different words, in order to be connected smoothly with the characters in front and behind.

On the Scale Design of the Network

A neocognitron usually consists of more layers and more neurons in each layer than any other neural network models in common use. Therefore, the factors which affect the size of the network is of great concern. Fukushima and his group have studied this problem [65], and their major conclusions are summarized as follows:

1. The sizes of spatial spread of connections between neural cells of two adjacent layers are determined by the complexity of the patterns to be tackled. If the complexity of the patterns is high, these sizes have to be small.

2. The optimal sizes of these spatial spread of connections are also affected by the degree of deformation which the neocognitron has to tolerate, especially in the lower stages. These sizes can be large if the probable deformation is small, but they must be small when large deformations are expected.

3. The number of cell planes in each stage of the network is determined by the number of pattern categories to be recognized. More specifically, it is equal to the number of different features to be extracted in each stage, and this is determined by the training pattern set.

4. The total number of stages of the network is determined by the complexity of patterns to be recognized. Complex patterns will result in more stages in the network because of the small sizes of the spatial spread of connections. Therefore, a larger number of stages are necessary to integrate information from all parts of the input pattern at the final stage of the network.

Similarity and Difference between Neocognitron and Le Cun's Network

The main idea in the design of the network's architecture is similar in neocognitron and Le Cun's network. Both utilize the scheme of partial connection to extract local features, and weight sharing for detecting the same feature at different locations. Also, both have an alternative arrangement of layers for feature extraction and positional error toleration.

The major difference between these two systems, however, lies in their training procedures. In a network developed under back propagation model, like that of Le Cun's group, training is implemented by propagating an output error signal all the way back through the network layer-wise and modifying the weight values accordingly. Back propagation is a supervised learning algorithm. Neocognitron, on the other hand, can be trained by either supervised or unsupervised learning, and the supervised training process in [65] is also different from the back propagation algorithm in three aspects. First of all, only part of the neurons in this network has variable input weight, the values of which need to be assigned through training. And instead of small random values, they are all initialized to zero. Secondly, training is performed in a forward direction, the same as a signal might go during its recognition. Thirdly, training is conducted in a non-iterative mode, that is, the training of a certain layer shall never start until those of all its preceding stages are finished.

Fukushima *et al.* also made an interesting comparison of the training time between their network and that of Le Cun *et al.* The outcome is 13 minutes on a SUN SPARCstation *versus* 3 days also on a SUN workstation. Nonetheless, the training

process of neocognitron needs a training set of patterns that are selected artificially. This is a skillful work, and the network's ability to recognize deformed characters depends ultimately on it.

Finally, the implementation of these two networks has reached similar recognition speed on handwritten numerals: 10 to 12 classifications per second by Le Cun's network with the help of a specially developed Digital Signal Processor board, and 8 characters per second by a neocognitron network realized on a parallel computer NCUBE.

1.3.4 Other Neural Network Models

Although back propagation network is presently the most popular neural model applied to character recognition, some other neural networks have been investigated for this purpose as well.

For instance, Wilson *et al.* developed an ART based method of feature extraction and classification [70], although later they too turned to the back propagation model [71]. This network has reached an accuray of above 99% on machine-printed digits and 80% on unconstrained hand-printed ones.

Also, Shimada's group proposes a new self-organizing method SOMA, abbreviated from Self-Organization to Modules by Activity propagation, to train multi-layer network [72]. SOMA is claimed to be a learning algorithm faster than back propagation due to the modularization of localized activation mechanism on hidden layers. A three layer network trained by SOMA is employed to recognize handwritten digits,

and a correction rate of 96% is achieved for testing data.

A pattern matching neural network is also addressed which matches an input to multiple candidates of the stored templates in parallel and finds out the best matching template, whose features are arranged in the same order as those of the input, regardless of positional differences between corresponding features [73]. The recognition rates are 98.2% for the learning set and 88.2% for the testing set when this network is empolyed to classify a group of handwritten numerals.

Besides, Alkon *et al.* have designed a neural network based on some neurobiological observations [74], [75]. It is called DYnamically STable Associative Learning (DYSTAL) network. Each output neuron in DYSTAL is connected, via patches, to a set of conditioned stimulus (CS) input elements and one of the unconditioned stimulus (UCS) input elements. These patches to model local learning are created or modified during the training by the associative presentation of the CS and UCS pattern. In testing, however, only the CS input patterns are provided. DYSTAL has already been applied in handwritten zipcode recognition [76], [77]. Either the bit-mapped digits or the features obtained through principal component analysis are used as CS patterns. In both cases, the best recognition rates recorded are in the range above 96%.

In addition to the above, associative memory networks are also considered. As described in [78], a special associative memory is proposed which has binary weights of its synapses. This makes it very suitable for hardware implementation using digital VLSI technology, and extremely fast in both storage and recollection phases even when simulated on sequential computers. An evaluation of the recognition ability of

associative memory model is made in [79]. In order to increase the system's robustness against the additive noise and shifting error, some techniques like defocusing preprocessing and the recursive use of the associative memory are discussed. The effectiveness of all these measures taken in these two networks is demonstrated experimentally in the recognition of letters of the English alphabet.

Further information about other neural network models suitable for pattern recognition can be found in the reviews of both [29] and [30].

1.4 Neural Networks in Chinese Character Recognition

Much research has already been conducted in the recognition of numerals and letters of the English alphabet using neural networks in the past five years. Some results look promising particularly in handwritten character recognition.

So far, however, the application of neural network approach in Chinese character recognition seems still at its beginning. At first, it is mentioned only for the purpose of illustrating the properties possessed by the newly proposed neural models. Under these circumstances, merely a very small number of computer generated low-resolution characters have been involved, as can be seen in [80] and [81]. The ideas are interesting and also instructive, yet nothing considered here is practical. Encouragingly, the reports of research work on Chinese character recognition under different neural network paradigms start to appear recently with real-life data in use.

1.4.1 Back Propagation Model

Although back propagation model has been very successful in handwritten digit recognition, whether it is applicable to Chinese character recognition remains to be seen, since there is no guarantee that the network can be trained in a reasonable finite amount of time. This is a critical issue because the training process of this model in a classification problem with a large variety of pattern classes may be too long to be of practical use.

Knowing the current capability of back propagation network, Mori *et al.* are trying to construct a large scale neural network for 3,000 handwritten Chinese character recognition by assembling many smaller back propagation networks, each of which is responsible for recognizing a small number of similar characters [59], [82]. The whole system is constructed with three kinds of functional units, that is, the SubNet which is an ordinary three layered back propagation network used to recognize a small set of characters, the SuperNet that is also a three layer feed forward network but larger than the SubNets and makes its decision by integrating the output from all the SubNets, and a network called OtherFilter, devised to improve the integration ability of the SuperNet. In this system, each SubNet is designed to recognize 9 characters, and a SuperNet can integrate the output of 30 SubNets. Therefore, totally 270 characters are involved in this group as part of the whole system. If 11 such groups of networks are integrated again by the same mechanism, the system will be capable of dealing with characters up to 3,000. Part of the system being able to recognize 270 Chinese characters has been constructed for computer simulation. The recognition

rate is above 93% for training samples, but drops to 74% when testing patterns are applied. Obviously, although this small scale network assembling approach can effectively reduce the number of characters faced by each single back propagation network, the function of integration issues a new challenge to the entire system.

Attempts to recognize similar Chinese characters with a back propagation network have also been made by others on both printed and handwritten characters. Kimura, for one, proposes a new back propagation learning algorithm based on the principle of minimizing not only the error but also the output variation to create consistent output for varied input patterns [83]. He tests his network with four similar characters, and it outperforms either the ordinary back propagation network or such a network trained with noisy patterns. Yen's group, on the other hand, apply a back propagation network to the recognition of thirty similar printed Chinese characters [84]. They design a noisy data relearning process, through which the robustness of the network is augmented, and the recognition rates above 95% have been reached for patterns with noise levels up to 20%.

The back propagation model is also used to recognize handwritten radicals of Chinese characters [85]. A modified logarithmic coordinate transform algorithm is employed to extract features invariant to both scaling and translation from the two dimensional radical image. Up to 96 radicals are involved in the experiments. Under some specified writing restrictions, the recognition rate of 80% has been achieved.

In all the above back propagation networks, the output layers have the same number of neurons as the number of classes they are currently handling. This is not

the case in the network of [86], the inputs and outputs of which are all binary vectors that represent the sequence of strokes of a hand-printed Chinese character and the corresponding Mandarin phonetic transcripts of the same character. In a network built to recognize 30 characters, only 14 output neurons are needed, and all the input characters have been classified correctly.

1.4.2 Single Layer Perceptron

In consideration of the complexity in both the amount and the variability of Chinese characters, and therefore the computational load required by a classifier made of back propagation network, Jeng's group employs a single layer perceptron instead [87] [89]. The features are extracted by retrieving boundaries of normalized character image and then quantizing these pixels to four possible orientations. The simulation is conducted on 100 hand-printed and 500 printed Chinese characters, and the latter contains six fonts. The recognition rates reported are above 98% and 99% respectively.

Although the number of printed characters recorded in this work is relatively large, the perspective of this netowrk to be a pattern classifier is still not convincing at this moment for the reasons cited:

- In the first place, the features extracted in this system are local ones which are more relevant to the character's structure rather than the style of its stroke. Therefore, the font irrelevance of the system relies on the feature extraction process instead of the neural network classifier. On the other hand, the effects of some geometric distortions which are often encountered in printed character

recognition like translation or rotation are neglected.

- Also, as indicated in [90], for a single layer network comprising a bank of Adalines, its statistical capacity approaches its deterministic capacity, which equals the number of inputs of each Adaline, as the number of Adalines becomes large. Although both the inputs and the outputs of the neurons on the output layer of the present network are not binary as those of an Adaline, it also needs to enlarge the dimensionality of its input vector, and therefore its capacity, to improve its performance. This requires more features which will be obtained by raising the resolution of cellular partition on character image, and eventually increase the sensitivity of this feature extraction process to the above-mentioned geometrical distortions.

- Finally, in addition to its capacity, the generalization ability of a neural network is a very important issue, for it implies directly to the classifier's capability in dealing with the input patterns in distorted forms. Widrow *et al.* made the following comment on this:

  > A network's capacity is of little utility unless it is accompanied by useful generalizations to patterns not presented during training. In fact, if generalization is not needed, we can simply store the associations in a look-up table, and will have little need for a neural network.

Unfortunately, the experiment performed in the work of Jeng's group can not verify the generalization ability of their neural network classifier. The testing

data used are characters of five fonts different from the training ones only in size. Since the character image should be normalized first in the feature extraction process, and the latter has been designed to reduce the effect of varying stroke widths, it can be expected that any such differences in this kind of testing data have already been eliminated before they are presented to the classifier.

1.4.3 Neocognition

Fukushima's Neocognitron enhanced with a retraining process and an inhibitory calibration is considered in [91]. This is a multi-layer network as well which recognizes Chinese characters based on their subpatterns, called primitives, of different levels of complexity. This network is noise tolerant, shift invariant and has a property of modularity which means it can be expanded to recognize more characters by simply adding and training some extra primitives needed to constitute new characters. The process of recognizing 20 Chinese characters is illustrated with greater detail. Nevertheless, due to the limitation of simulation time and memory space, character sets with more than 50 elements are not allowed by the present system.

Also, Fukushima's group applies the selective attention model built on neocognitron to Chinese character recognition [69]. The radicals of which the Chinese characters are composed are employed as individual patterns during the training phase. An auxiliary circuit is added to the network to detect the pattern's center of gravity, since the information on not only the shape of a radical but also its position, left hand side, right hand side or, upper or lower parts, is required. A preliminary simulation work

is conducted with only eight radicals, and sixteen character samples which contain nine different characters are shown to be recognized successfully.

The strokes of the characters used in the above two systems are all one pixel wide. But the selective attention model has already shown the ability to recognize characters varied in a way of handwritten nature.

1.4.4 Associative Memory Network

Yao designed a system of content addressable associative memory based on Hopfield model, which is supposed to perform the task of Chinese character recognition [92], [93]. He described as well the circuit of the network assumed to do the job in place of a digital computer. Unfortunately, no results on any recognition experiments are reported in his papers.

1.4.5 Other Networks

The DYSTAL Network DYSTAL has been used to classify not only handwritten digits (See Section 1.3.4) but also handwritten Chinese characters [76], [77]. The features utilized are obtained through principal component analysis [94]. 40 different characters are involved in a preliminary experiment which yields nearly 90% correct classification.

The Combination of Networks Because it is infeasible to solve the problem of Chinese character recognition with one large scale neural network of any single

model, Iwata *et al.* propose a four layer network, called CombNET, which is in fact a combination of networks of existing models [95]. CombNET is in a comb structure. It has a vector quantizing network to form its first layer as the stem, and then the same number of three layer back propagation networks as that of the neurons in the vector quantizing network to be its branches. Its classification strategy is that of the two stage approach discussed in Section 1.2.2. The vector quantizing network partitions the whole character set into many subgroups, characters in each of which will be subsequently classified by a corresponding back propagation network. The network has been used to recognize 2965 printed Chinese characters with a reported recognition rate of 99.5%. However, both the formation of the input patterns and the difference between the training and testing data are not described explicitly. As a result, this network's applicability still remains open due to the concerns similar to those addressed in the second part of this section.

1.5 Neural Networks in Feature Extraction

Neural networks are mostly used as classifiers in pattern recognition at present. Yet Graf and Jackel *et al.* [96], [97] have designed a neural net chip especially for feature extraction and even character skeletonization, and it has already been used in a digit recognition system [98], [99]. The idea is rather instructive in spite of its template matching nature, which is intrinsically a computationally intensive operation of many components, low-accuracy, vector dot products, suitable for implementation with parallel analog chips.

Besides, Zhu and her colleagues have developed a multi-layer neural network which consists of two parts — one for feature detection and the other for pattern classification [100], [101]. It is designed to follow the functions of the retina and the neuron cells in the brain of a living body separately. Although its classification part is just a back propagation network, the design of its first few layers for feature detection is really specific. It uses different sorts of neurons such as on-center cell, off-center cell *etc.* to detect some local features like end point, cross point, and so on. In a computer simulation on handwritten numeral and capital Roman alphabet recognition, it has reached a recognition rate above 90%. This network, however, is not the only one that can carry out feature extraction. As a matter of fact, both Le Cun's network and Fukushima's neocognitron have this property as well.

Also, Rubner and Schulten propose a network which can perform principal component analysis [102]. This is a two-layer network with lateral connections in the output layer. Hyman *et al.* apply this network to extract the principal components from a handwritten Chinese character image as the features for its further classification [94]. In a principal component analysis network built for 956 handwritten Chinese characters, training continues until the first 160 principal components converge. This accounts for . . s than 5% of that of the original image pattern in terms of vector's dimensionality, which is 4032 (63×64). And a simulation on the classification of 40 characters with features extracted thereof reached nearly 90% successful recognition.

1.6 Objectives and Organization of this Thesis

There is no need to be surprised that up to date the research on application of neural networks to recognize Chinese characters is still at its beginning. The theory of artificial neural networks itself is far from well-established and complete at present, since our brain, the biological model of artificial neural networks, is extremely complicated and our knowledge about it is still fairly limited. However, it has offered some solutions to so many pattern recognition problems such as that of Chinese characters so well that there is no doubt that neural networks which are attempted to simulate the behavior of human brain, will ultimately play an important role in solving such problems artificially.

In this thesis, a novel neural network classifier will be explored based on the model of associative memory network. It is developed with an application background in the recognition problems of large number categories, such as Chinese characters. In Chapter 2, the applicability of associative memory network as a pattern classifier is discussed first. After revelation of the intrinsic similarity between associative mapping and the pattern recognition process, the basic problem in the construction of associative memory classifier, *i.e.* to find the suitable output vectors, or inner codes, associated with the set of given input patterns to be recognized in accordance with their characteristics, is addressed. This process will be known as inner coding. Two types of measurements, the average correlation coefficient and the probability distribution of input pattern's components, are established on the statistical basis to describe these characteristics. The selection of output vectors is discussed by an ex

ample of a set of real-life multi-font Chinese characters, and Hadamard vectors are chosen for a series of their desired features which result in a success in the construction of such a neural network classifier.

Chapter 3 will discuss the inner coding of associative memory classifier in more detail. Its purpose is to search for better inner coding schemes for the sake of improvements in the network's classification performance. Some theoretical aspects of this problem are addressed first, such as the evaluation of associative recollection process and the criterion to judge optimal inner coding schemes, their existence, and the complexity of optimal coding scheme seeking through enumeration, *etc.* Next to these is proposed an inner coding scheme seeking strategy applicable to practical situations. The efficacy of this strategy is demonstrated by a further discussion on the recognition of the same set of Chinese characters used in the former chapter when Hadamard vectors have been selected as inner codes in an associative memory classifier. An optimal coding strategy will be proved under some ideal conditions, a remedial measure will be taken to apply this strategy to real-life data, and improvements in network's behaviour will be observed in computer simulation.

Technically, another major part in a pattern recognition system, in addition to pattern classifier, is the feature extractor. The success of the entire system depends on not only the effectiveness of each of them, but also their operation in concert. Therefore, the corresponding feature extraction method when associative memory network is used as pattern classifier will be discussed in Chapter 4. Addressed firstly there is a change in the fundamental objectives of feature extractor utilized in coorperation

with a neural classifier. A feature extraction method is proposed thereafter which can detect features suitable for an associative memory classifier. Such features are also invariant to some common geometrical distortions, which is desirable in printed character recognition. Details of this technique are studied. Its efficacy in the improvement of the feature vector's suitability to our neural classifier and further, the effectiveness of the entire system, including its robustness tested by a set of data with alike characters as the intentionally introduced similarities in it, are manifested with experimental results.

The above studies are our preliminary work on the application of associative memory network as a pattern classifier to solve the recognition problems of large number categories. Some of these materials have already appeared in our publications before [103]-[109].

Finally in Chapter 5 is the conclusions of this thesis.

Chapter 2

The Applicability of Associative Memory as Pattern Classifier

2.1 Introduction

Neural networks have already been used widely in alphanumeric recognition, especially in handling handwritten characters. So far, the most successful models applied seem to be the back propagation algorithm, like Le Cun's network [37]-[41], and Fukushima's neocognitron [64], [65]. These two models have also been tried for Chinese character recognition [59], [69], [82]-[86], [91]. However, the number of categories of Chinese characters is of orders of magnitude larger than that of alphanumeric ones, not to mention their variability and complexity in structure. At present, the convergence and the speed of the training process may cripple a back propagation network when the classes grow. Similarly, the scale of the network and the skillful or even tricky training pattern design which is critical to the network's final performance can

limit the increase of patterns to be recognized by the neocognitron.

Besides, in addition to their fast computational nature, the major motive to use neural networks as pattern classifiers stems from their resistance to noise and other distortions in the recognition process. Therefore, when the incoming patterns shall inevitably be subject to deformations before its recognition, which are usually the cases of actuality, neural networks are particularly useful. However, if the forms of pattern data are really stable, then neural networks are unnecessary, some traditional classification methods, such as template matching *etc.*, may be already good enough to solve the problem.

In this chapter, the applicability of another neural network model as pattern classifier will be investigated. The model is associative memory network, and an associative memory classifier is developed for its potential to tackle the recognition problems containing a large number of categories, such as that of Chinese characters. Of course, the data to be dealt with include those that are contaminated heavily. In the next section, a comparison of associative mapping and pattern recognition process is made, followed by a brief description of associative memory networks in Section 2.3. Sections 2.4 and 2.5 discuss thoroughly the construction of an associative memory classifier, taking the recognition of a set of multi-font Chinese characters as an example. The effectiveness of this novel neural network classifier is verified in Section 2.6 by computer simulation.

2.2 Pattern Recognition Property of Associative Memory

A memory usually involves a storage mechanism which utilizes a storage medium and some operations on it, called memory functions. In an associative memory, many pattern pairs, each having a stimulus and a response pattern, are stored associatively in advance. Then, when an input stimulus signal is applied to the memory system where it is pre-stored, the pattern which has been memorized associatively with the input will be generated automatically. This is the so-called associative recall process, see Figure 2.1(a). Obviously, this associative retrieval process can be taken as a signal transfer operation in a physical media, whereby an input pattern, also called the key or, the associative cue, *etc.*, is directly transformed into a corresponding output pattern, the recollection.

On the other hand, in a general sense, a visual pattern recognition process, such as the recognition of characters, is nothing but a technique to associate a symbolic identity with the image of an object, and this category identity can also be coded, see Figure 2.1(b). Therefore, it is evident that the pattern classification process can be regarded as a special case of associative recollection by simply assigning the input signal to the memory as pattern vectors to be recognized and its output the category codes, or more generally, any form of inner codes in an information system, so that the machine knows what the coming pattern is, what it means *etc.* As a matter of fact, the idea of associative recall itself is partly inspired by the ability of human beings in the recognition of handwriting and speech with major variations, distortions, and even omissions.

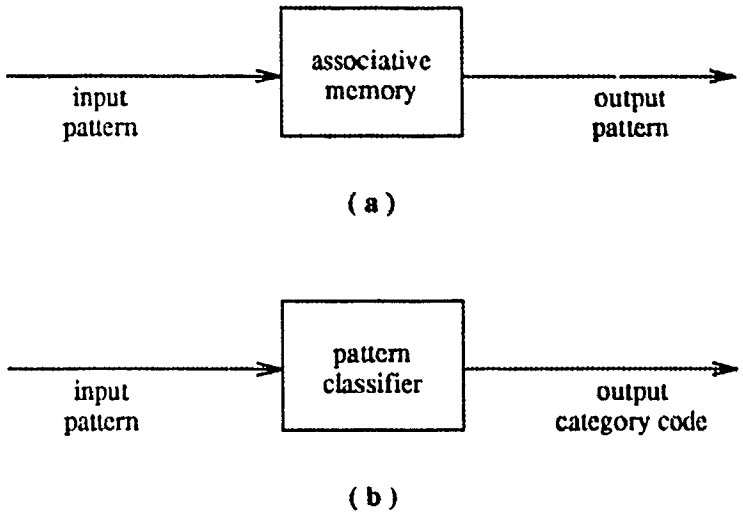


Figure 2.1: The information transform of an associative memory (a), and a pattern classifier (b).

All in all, in a logical sense, pattern classification may be taken as an associative mapping from the pattern space to a corresponding one of category. By taking all the intrinsic mechanism on the transform of information as a black box, the similarity between a pattern classifier and an associative memory is quite obvious. And that explains as well why the associative memory model is selected in our exploration of novel neural network pattern classifiers.

Incidentally, associative memory model has the following two additional characteristics which are both particularly desirable for a pattern recognition system:

1. It stores information in a distributed, robust manner. Local damages to the internal structure of the memory network might cause declined accuracy in overall responses, but not a total loss in certain respects of the storage.

2. It can regenerate the correct response pattern even though the input stimulus pattern is distorted or incomplete. That is to say, the limited amounts of distortions or omissions in an input pattern may cause little or no deformation in the associated output pattern, and increasing amounts of distortions in the stimulus will cause correspondingly larger deformations in the response with no abrupt breakdown of recollection.

2.3 Associative Memory Network

2.3.1 Network Structure

In the structural perspective, the fundamental form of the associative memory is a one layer neural network (except the input layer), the one which also serves as the network's output layer, see Figure 2.2. The signal to each node of the input layer is the corresponding component of input vector, and those from the output layer form the output pattern of associative recall. Every pair of nodes from different layers is connected under a certain weight. This is where the power of associative memory network lies. The associative memory network in [110] and [111] is discussed just under this structure

There are also other specific forms of associative memory. The most important one is the Hopfield net, which for the first time introduces the nonlinear feedback dynamics into associative memory network [112], [113]. The architecture of the Hopfield net is illustrated schematically in Figure 2.3. It has only one layer of neurons, all are

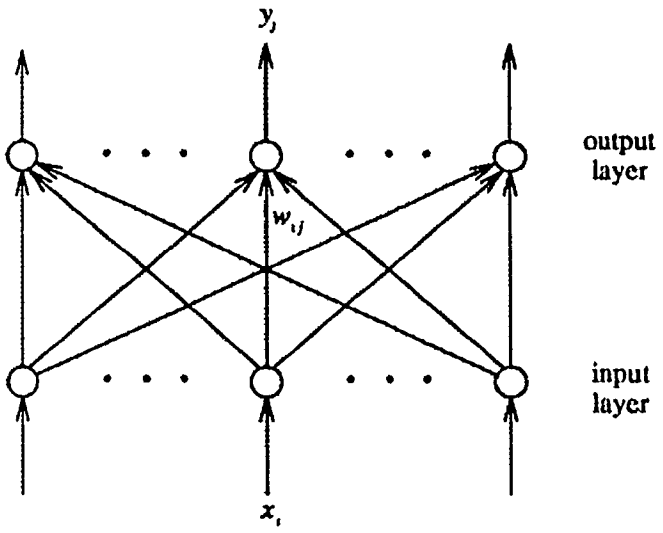


Figure 2.2: The fundamental configuration of associative memory network.

connected laterally to all others. After the network is stimulated by an input, the signals within the network will evolve gradually in the nonlinear feedback manner. Finally, they will settle on a fixed pattern, and that is the system's output.

In addition, Kosko goes a step further by extending this dynamic mechanism to a two layered network, called bidirectional associative memory, for paired data associations [114], [115], the schematic diagram of which is shown in Figure 2.4. There are two different connections, and therefore two different weights, between every pair of neurons from each of the two layers separately.

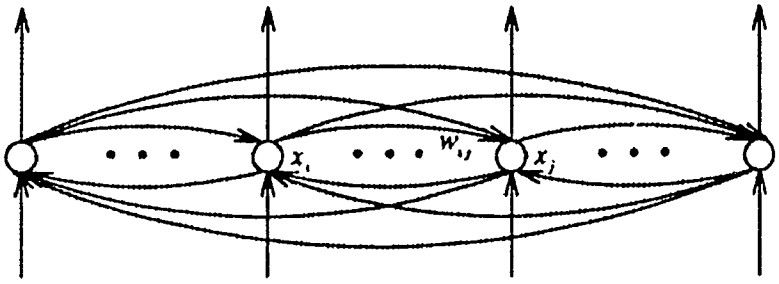


Figure 2.3: The Hopfield network.

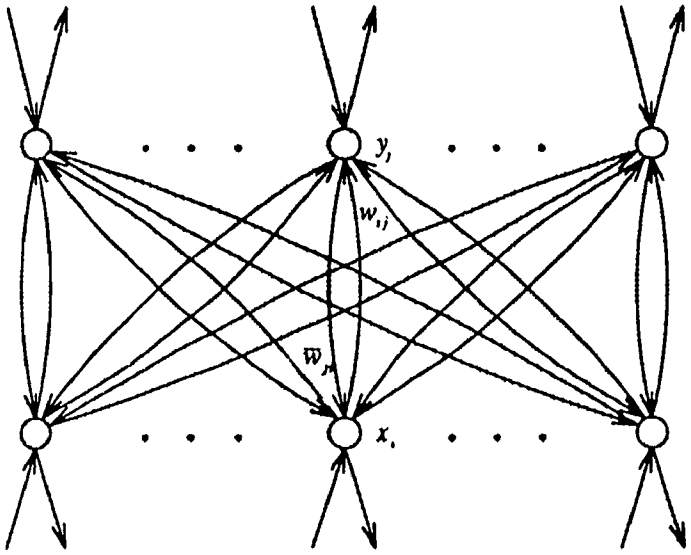


Figure 2.4: The bidirectional associative memory network.

2.3.2 Mathematical Description

From the mathematical point of view, all the weights on the connection paths of associative memory networks can be represented together in matrix form.

Suppose $\{(\vec{x}_k, \vec{y}_k)|k = 1, 2, \cdots, s\}$ is a set of associated pattern pairs, where $\vec{x}_k = (x_{k1}, x_{k2}, \cdots, x_{km})$, $\vec{y}_k = (y_{k1}, y_{k2}, \cdots, y_{kn})$ are row vectors in m- and n-dimensional spaces X and Y respectively. For each of these pattern pairs, a matrix M_k is obtained through the outer product of $\vec{x}_k$ and $\vec{y}_k$:

$$\begin{aligned} M_k &= \vec{x}_k^T \vec{y}_k \\ &= \begin{pmatrix} x_{k1} \\ x_{k2} \\ \vdots \\ x_{km} \end{pmatrix} (y_{k1}, y_{k2}, \cdots, y_{kn}) \\ &= \begin{bmatrix} x_{k1}y_{k1} & x_{k1}y_{k2} & \cdots & x_{k1}y_{kn} \\ x_{k2}y_{k1} & x_{k2}y_{k2} & \cdots & x_{k2}y_{kn} \\ \vdots & & & \\ x_{km}y_{k1} & x_{km}y_{k2} & \cdots & x_{km}y_{kn} \end{bmatrix} \end{aligned} \tag{2.1}$$

where $\vec{x}_k^T$ represents a column vector, the transpose of $\vec{x}_k$. Summing these matrices up, gives:

$$M = \sum_{k=1}^{s} M_k = \sum_{k=1}^{s} \vec{x}_k^T \vec{y}_k$$

$$= \begin{bmatrix} \sum_{k=1}^{s} x_{k1}y_{k1} & \sum_{k=1}^{s} x_{k1}y_{k2} & \cdots & \sum_{k=1}^{s} x_{k1}y_{kn} \\ \sum_{k=1}^{s} x_{k2}y_{k1} & \sum_{k=1}^{s} x_{k2}y_{k2} & \cdots & \sum_{k=1}^{s} x_{k2}y_{kn} \\ \vdots & & & \\ \sum_{k=1}^{s} x_{km}y_{k1} & \sum_{k=1}^{s} x_{km}y_{k2} & \cdots & \sum_{k=1}^{s} x_{km}y_{kn} \end{bmatrix} \tag{2.2}$$

and this is the associative memory where s pattern pairs are stored.

The information recollection with the associative cue $\vec{x}_k$ can be attained by its multiplication with memory M:

$$\vec{x}_k M = \vec{x}_k \sum_{l=1}^{s} \vec{x}_l^T \vec{y}_l = \vec{x}_k \vec{x}_k^T \vec{y}_k + \sum_{l \neq k} \vec{x}_k \vec{x}_l^T \vec{y}_l \tag{2.3}$$

If the input vectors $\vec{x}_1, \vec{x}_2, \cdots, \vec{x}_s$ are orthonormal, namely:

$$\begin{aligned} \vec{x}_k \vec{x}_k^T &= 1 \\ \vec{x}_k \vec{x}_l^T &= 0 \qquad , \qquad \forall\, l \neq k \end{aligned} \tag{2.4}$$

then Eq.(2.3) gives:

$$\vec{x}_k M = \vec{y}_k \tag{2.5}$$

which is a perfect recall.

If, however, $\vec{x}_1, \vec{x}_2, \cdots, \vec{x}_s$ are not orthogonal, as in general cases, the second term on the right-hand side of Eq.(2.3) is a noise term which introduces crosstalk to contaminate the desired retrieval pattern, even though the key applied is of the exact form of a pre-stored pattern.

This is the most basic form of associative memory, it is actually a linear memory.

To make the memory system more robust to noise, such as crosstalk, and more importantly, those generated when a distorted input signal is applied, nonlinearity has been introduced into associative mapping by letting recalled output $\vec{y}$ be some nonlinear transformation F of the product $\vec{x}M$:

$$\vec{y} = F(\vec{x}M) \tag{2.6}$$

The simplest form of F is a threshold function, for instance:

$$y_i = \begin{cases} 1, & \text{if } \vec{x}M^i > 0 \\ -1, & \text{if } \vec{x}M^i < 0 \end{cases} \tag{2.7}$$

under bipolar coding, where y_i is the ith component of the recollected vector $\vec{y}$, and M^i is the ith column of memory matrix M. In this thesis, unless specified otherwise, the nonlinear transformation will always take the form of this function hereafter.

Nonlinear transformation, in the first place, can lead to a perfect recollection initialized by $\vec{x}_k$ for a set of orthogonal input patterns which need not be normalized. Since in this case:

$$\vec{x}_k M = \vec{x}_k \vec{x}_k^T \vec{y}_k$$

and obviously:

$$\vec{x}_k \vec{x}_k^T > 0$$

but may not equal to 1 as required in Eq.(2.4), therefore:

$$\vec{y} = F(\vec{x}_k M) = \vec{y}_k \tag{2.8}$$

More importantly, according to Eq.(2.3), the output of an associative retrieval process activated by input $\vec{x}$ is a combination of stored patterns $\vec{y}_k$ $(k = 1, 2, \cdots, s)$

in space Y, each multiplied by a factor $\vec{x}\vec{x}_k^T$. If $\vec{x}$ is close enough to a pre-stored pattern $\vec{x}_k$, then it is reasonable to expect that each $\vec{x}\vec{x}_l^T$ $(l = 1, 2, \cdots, s,\ l \neq k)$ is much smaller than $\vec{x}\vec{x}_k^T$. Therefore, the sign of $\vec{x}M'$ will not be changed in the summation by the corresponding component of the second term on the right-hand side of Eq.(2.3) quite often, that is, a correct value of y_i will be obtained most probably after the nonlinear transformation, and so is $\vec{y}$. In this way, the whole memory system is made less sensitive to noise caused by crosstalk and distorted input signals.

When spaces X and Y are identical, the network is regarded as an autoassociative network. In this case, if the mapping result $\vec{x}'$ of an input vector $\vec{x}$ is not so satisfied, that is, if it is not near enough to a certain pre-stored pattern $\vec{x}_k$, it can be fed back into M again and again, and ideally, a perfect recall would be achieved finally. Kosko extends the idea of nonlinear feedback to the heteroassociative memory where associated pairs $(\vec{x}, \vec{y})$ are from two different spaces X and Y, especially the case when their dimensionalities are unequal. Suppose $\vec{y}' = F(\vec{x}M)$, and the feedback trial is continued by generating $\vec{x}'$ from $\vec{y}'$ simply through M^T, the transpose of memory matrix M. Therefore,

$$\vec{x}' = F(\vec{y}'M^T)$$

and then,

$$\vec{y}'' = F(\vec{x}'M)$$

$$\vec{x}'' = F(\vec{y}''M^T)$$

and so forth, until the final result, a stable pair $(\vec{x}_f, \vec{y}_f)$, is reached in the following manner:

$$\vec{y}_f = F(\vec{x}_f M)$$

and

$$\vec{x}_f = F(\vec{y}_f M^T)$$

2.4 Construction of Associative Memory Classifier

2.4.1 Basic Problem in the Construction of a Pattern Classifier from Associative Memory

One of the first application of the associative memory network is information recovery. Kohonen furnishes an example of this by illustrating a system to memorize some human facial pictures [111]. These pictures can be recollected with the retrieval keys either superimposed with white noise or masked. The latter case is in fact the recollection of the whole pattern based on partial information. Evidently, under this circumstance, both patterns in each associated stimulus-response data pair are well defined beforehand.

Now it has been known that a pattern classification process can be realized through associative mapping and, the stimulus and response signal of each associated data pair are the pattern to be recognized and its corresponding category. A pattern to be recognized, whether it is the original signal representing an object itself or any of its features specifically extracted, usually takes the form of vectors, and is well defined in advance. Such patterns will be called input patterns or input vectors *etc* hereafter. On the other hand, the category to which a pattern belongs is itself nothing but a logical identity. In order to apply the associative memory network as a pattern

classifier, this logical symbol needs to bear a concrete form in terms of signal pattern, that is to say, a specific code or vector pattern should be assigned to it. This is actually a coding process of these category symbols. These category codes are merely for the internal use of an information system. Consequently, they will be referred to as inner codes or, in the network perspective, output patterns or output vectors. Correspondingly, this process will be called inner coding.

In brief, when an associative memory is to be utilized as a pattern classifier, unlike the case of its other application such as information recovery, the form of its output patterns is not pre-defined, and therefore open for our selection. In the view of identification, this output pattern coding is arbitrary, subject to no specific requirements, since there exists no effect of a particular form of the inner codes on their logical meanings. However, inner codes are the response patterns to be associated with their corresponding stimuli. Since an associative memory functions on the basis of connection weights of its neurons, and these weights are composed of every stimulus-response pair it memorizes, inner codes selection may have an influence on its behaviour of information retrieval. For this reason, it should be addressed specifically and dealt with carefully. As a matter of fact, inner coding turns out to be the fundamental problem in the construction of a pattern classifier from associative memory network, called associative memory classifier, as it can be seen later on.

2.4.2 Inpact of Inner Coding on the Performance of Associative Memory Classifier

The input patterns play a principal role in the weight formation of an associative memory. Even with the same set of inner codes, a system of different input patterns may behave differently. On the other hand, some specific groups of input patterns may allow the variance of inner codes within a certain range to achieve the same classification effect. However, the input patterns of an associative memory classifier are usually well defined in advance. Therefore, the only room left for the flexibility of its weight formation is that in the selection of the corresponding output vectors, or inner codes. To get the general picture about the impact of inner coding on the performance of an associative memory network, a discussion over some special cases of these patterns is really helpful.

Assume $\{\vec{x}_k | k = 1, 2, \cdots, s\}$ is a set of m-dimensional input patterns, and their associated inner codes $\{\vec{y}_1, \vec{y}_2, \cdots, \vec{y}_s\}$ are n-dimensional. Also, both of them in an associated pair are binary patterns coded bipolarly, *i.e.* each component of both $\vec{x}_k$ and $\vec{y}_k$ will be either 1 or -1. Therefore:

$$\begin{aligned} \vec{x}_k \vec{x}_k^T &= (x_{k1}, x_{k2}, \cdots, x_{km}) \begin{pmatrix} x_{k1} \\ x_{k2} \\ \vdots \\ x_{km} \end{pmatrix} \\ &= x_{k1}^2 + x_{k2}^2 + \cdots + x_{km}^2 \qquad (2.9) \\ &= m \end{aligned}$$

The memory network to store them is M:

$$M = \sum_{k=1}^{s} \vec{x}_k^T \vec{y}_k$$

If all these input vectors are identical, that is:

$$\vec{x}_1 = \vec{x}_2 = \cdots = \vec{x}_s$$

Then:

$$\vec{x}_k \vec{x}_1^T = \vec{x}_k \vec{x}_2^T = \cdots = \vec{x}_k \vec{x}_s^T = m \quad , \qquad k = 1, 2, \cdots, s$$

And:

$$\begin{aligned} \vec{x}_k M &= \vec{x}_k \sum_{l=1}^{s} \vec{x}_l^T \vec{y}_l \\ &= \sum_{l=1}^{s} \vec{x}_k \vec{x}_l^T \vec{y}_l = \sum_{l=1}^{s} m \vec{y}_l \qquad k = 1, 2, \cdots, s \\ &= m \sum_{l=1}^{s} \vec{y}_l \end{aligned}$$

Hence, the retrieval result $\hat{\vec{y}}_k$ of associative cue $\vec{x}_k$ is:

$$\hat{\vec{y}}_k = F(\vec{x}_k M) = F(m \sum_{l=1}^{s} \vec{y}_l) = F(\sum_{l=1}^{s} \vec{y}_l) \quad , \qquad k = 1, 2, \cdots, s$$

Therefore:

$$\hat{\vec{y}}_1 = \hat{\vec{y}}_2 = \cdots = \hat{\vec{y}}_s$$

This means that if the input patterns are identical, then no matter what kind of inner codes is employed, they are still indistinguishable. This result is reached irrelevant to any particular shape $\vec{y}_k$s may take.

It is on this basis that the following claim is reached:

Observation 1 *Inner coding may not be able to improve the distinctiveness of the original input patterns.*

Now let $\{\vec{x}_1, \vec{x}_2, \cdots, \vec{x}_s\}$ be a set of orthogonal vectors. Then according to Eq (2.8):

$$\hat{\vec{y}}_k = F(\vec{x}_k M) = \vec{y}_k$$

Therefore, the identification of $\vec{x}_k$s will depend on the inner codes selected. For instance, if the vectors in the output pattern set are all pairwise different, then these input patterns are definitely distinguishable, and this distinctiveness of input patterns can be kept under any different sets of output vectors as long as they possess such a property. But if any two of these output patterns are identical, then the two corresponding input patterns can never be differentiated from each other any more.

This implies another claim as below:

Observation 2 *For a certain group of input patterns, different set of inner codes may cause different recognition behaviour of an associative memory classifier.*

The above are some extreme cases. Now, let us check the impact of inner coding in another instance.

Presume $\{\vec{x}_k | k = 1, 2, \cdots, s\}$ is a set of m-dimensional patterns that possess the property:

$$\begin{aligned} \vec{x}_k \vec{x}_l^T &= x_{k1}x_{l1} + x_{k2}x_{l2} + \cdots + x_{km}x_{lm} \\ &= \frac{1}{2}m \qquad (2.10) \\ &\forall\ k, l = 1, 2, \cdots, s,\ \wedge\ k \neq l \end{aligned}$$

Since the length of vector $\vec{x}_k$ is:

$$||\vec{x}_k|| = \vec{x}_k \vec{x}_k^T = m \quad , \qquad k = 1, 2, \cdots, s$$

Therefore:

$$\begin{aligned} \vec{x}_k \vec{x}_l^T &= \frac{1}{2}||\vec{x}_k|| \\ &= \frac{1}{2}||\vec{x}_l|| \quad , \qquad \forall \ k, l = 1, 2, \cdots, s \ \wedge \ k \neq l \end{aligned}$$

This means that $\{\vec{x}_1, \vec{x}_2, \cdots, \vec{x}_s\}$ is a set of vectors the projection of each to any other one is a vector in the same direction as the latter but only half of the length.

In a hyperspace with its dimensionality no less than s ($s \leq m$), this can be a set of vectors represented by the s edges of an s-dimensional hyper regular tetrahedron drawn out of one and the same vertex. Such a case which can be accommodated in a three dimensional space is shown in Figure 2.5. Under this circumstance, there can be at most three such vectors, for instance, $\vec{OA}$, $\vec{OB}$, and $\vec{OC}$. Vectors $\vec{OA'}$ and $\vec{OB'}$ are the projections of $\vec{OC}$ on $\vec{OA}$ and $\vec{OB}$ respectively.

Now suppose s is the multiple of both 2 and 3, and this set of patterns forms an associative memory M with a set of associative output patterns $\{\vec{y}_1, \vec{y}_2, \cdots \vec{y}_s\}$. Also notice that according to Eq.(2.3) each component of the n-dimensional $\vec{x}_k M$ has:

$$\begin{aligned} (\vec{x}_k M)_i &= \vec{x}_k M^i = \vec{x}_k \sum_{l=1}^{s} \vec{x}_l^T y_{l_i} \\ &= \vec{x}_k \vec{x}_k^T y_{k_i} + \sum_{l \neq k} \vec{x}_k \vec{x}_l^T y_{l_i} \\ &= m y_{k_i} + \frac{1}{2} m \sum_{l \neq k} y_{l_i} \end{aligned} \tag{2.11}$$

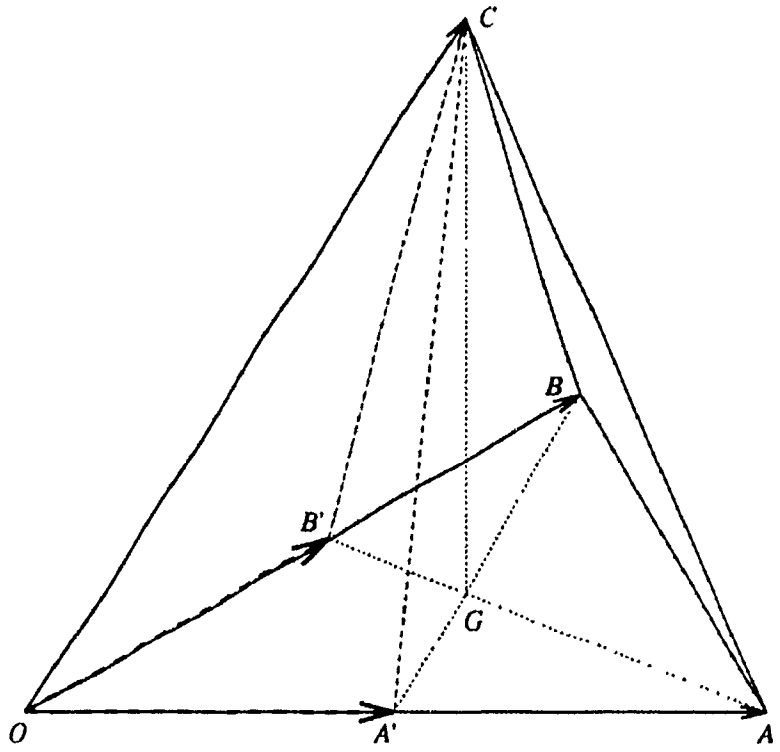


Figure 2.5: A set of input vectors represented by the edges of a regular tetrahedron sharing the same vertex.

Consequently, the ith component ($i = 1, 2, \cdots, s$) of the recollected pattern is:

$$\begin{aligned} \hat{y_{k_i}} &= F(\vec{x}_k M^t) = F(m y_{k_i} + \frac{1}{2} m \sum_{l \neq k} y_{l_i}) \\ &= F(y_{k_i} + \frac{1}{2} \sum_{l \neq k} y_{l_i}) \end{aligned} \tag{2.12}$$

It can be seen then that in this case different inner coding schemes will lead to different associative retrieval results with the ith component of the output pattern as an example.

For instance, if y_{l_i}s ($l = 1, 2, \cdots, s$) are coded in a way that half of them are 1, while the other half are -1, then clearly:

$$\frac{1}{2} \sum_{l \neq k} y_{l_i} = -\frac{1}{2} y_{k_i} \quad , \qquad k = 1, 2, \cdots, s \tag{2.13}$$

regardless of the sign of y_{k_i}. Therefore, according to Eq.(2.12):

$$\begin{aligned} \hat{y_{k_i}} &= F(y_{k_i} + \frac{1}{2} \sum_{l \neq k} y_{l_i}) \\ &= F(y_{k_i} - \frac{1}{2} y_{k_i}) \\ &= F(\frac{1}{2} y_{k_i}) \qquad\qquad k = 1, 2, \cdots, s \\ &= y_{k_i} \end{aligned} \tag{2.14}$$

Namely, this component of all the output patterns can be recollected correctly.

However, if inner coding has been conducted so that only one third of output patterns' ith component share the same value, say 1, while the others -1, then:

$$\begin{aligned} y_{k_i} + \frac{1}{2} \sum_{l \neq k} y_{l_i} &= 1 + \frac{1}{2}[(\frac{1}{3} s - 1) \cdot 1 + (\frac{2}{3} s) \cdot (-1)] \\ &= -\frac{1}{6}(s - 3) \qquad\qquad \forall \; y_{k_i} = 1 \end{aligned}$$

$$y_{k_i} + \frac{1}{2}\sum_{l\neq k} y_{l_i} = -1 + \frac{1}{2}[(\frac{1}{3}s)\cdot 1 + (\frac{2}{3}s - 1)\cdot(-1)] \qquad (2.15)$$
$$= -\frac{1}{6}(s+3) \quad , \ \forall \ y_{k_i} = -1$$

Since $s \geq 6$, both $s+3$ and $s-3$ are positive. Hence, no matter what kind of sign the current y_{k_i} has, Eq.(2.15) indicates:

$$\hat{y_{k_i}} = F(y_{k_i} + \frac{1}{2}\sum_{l\neq k} y_{l_i}) = -1 \quad , \qquad k = 1, 2, \cdots, s \qquad (2.16)$$

That is to say, only two thirds of these ith components can be recollected properly.

Based on the above discussions, the impact of pattern forms (both input and output), and therefore of inner coding on the performance of associative memory classifier can be summarized briefly as below:

Conclusion 1 *The recognition performance of an associative memory classifier relies ultimately on the distinctiveness of its input patterns. Inner coding can influence its functioning, and an appropriate selection of inner codes may result in reaching or approaching, but not overstepping, the limit of an associative memory's classification ability.*

2.4.3 Measurements of Pattern Vectors' Fitness to Associative Memory Classifier

Once an associative memory has been configured, its classification performance will be determined by the data in use, *i.e.* the inputs and their associated outputs of this associative memory classifier. A set of output vectors with desirable properties can

help the associative memory classifier to function properly and reach its discriminating ability, which is determined intrinsically by the suitability of input vectors to this neural network, as far as possible. Therefore, certain characteristics of an associative memory classifier's input vectors, which may be in the original form of patterns to be recognized or their extracted feature vectors, play the decisive role in the success of this recognition system, and the set of output vectors can only be selected as appropriate accordingly.

To inspect the fitness of a set of input vectors to associative memory classifier, some effective measurements should be established. In Section 2.3.2, Eq.(2.2) shows that the information storing raises no requirements on the characteristics of input and output patterns. But these do affect the network's classification behaviour according to Eq.(2.3).

Apparently, the first term on the right-hand side of Eq.(2.3) contains the expected recollection result. The second one, however, is the noise introduced by crosstalk with the other associated pattern pairs stored in this one and the same memory M. Since the disturbance caused by each $\vec{y}_l$ $(l \neq k)$, when $\vec{y}_k$ is to be retrieved, is factored by $\vec{x}_k \vec{x}_l^T$, the correlation of input vectors is of great significance.

Definition 1 *The correlation coefficient of any two m-dimensional bipolar vectors* $\vec{x}_k$ *and* $\vec{x}_l$ *is:*

$$\begin{aligned} \mu_{kl} &= \frac{|\sum_{i=1}^{m} x_{ki} x_{li}|}{m} \\ &= \frac{|\vec{x}_k \vec{x}_l^T|}{m} \end{aligned} \tag{2.17}$$

Obviously,

$$\mu_{kl} = 0 \ , \qquad \text{if} \quad \vec{x}_k \perp \vec{x}_l \tag{2.18}$$

and

$$\mu_{kk} = 1 \tag{2.19}$$

Definition 2 *Suppose* $\{\vec{x}_k | k = 1, 2, \cdots, s\}$ *is a set of binary vectors coded bipolarly. Then the average correlation coefficient of the whole set of vectors is:*

$$\mu = \frac{1}{s(s-1)/2} \sum_{k<l} \mu_{kl} \quad , \qquad k, l = 1, 2, \cdots, s \tag{2.20}$$

In line with the above definitions, the average correlation coefficient of a set of vectors will vary within the range from 0 to 1. The smaller this number is, the more probable the vectors of this group are orthogonal to one another. Clearly, it would be advantageous if a group of pattern vectors are in such a form that all the correlation coefficients μ_{kl} will approach 0 as close as possible, due to the fact that during the associative retrieval the crosstalk introduced are proportional to these factors:

$$\vec{x}_k \vec{x}_l^T = \pm n \mu_{kl} \tag{2.21}$$

Therefore, μ, the average correlation coefficient over the entire set of bipolar vectors, is a meaningful measurement on the orthogonality of these vectors in a statistical sense.

In the application with real-life data, the absence of a limitation on the trend for μ_{kl}s to approach 0 seems impossible, for the extreme case in which all the μ_{kl}s are 0 simply implies that such a set of vectors is mutually orthogonal. Thus, in dealing

with a practical problem, the existence of crosstalk is inevitable. Nonetheless, their effect can be reduced by counterbalancing them through the proper selection of a set of associated output vectors. That is to say, with a group of inner codes $\vec{y}_l$s chosen specifically, the value of the second term in Eq.(2.3) will be self-depressed so that it is small enough not to override that of its first term:

$$|\sum_{l\neq k}\vec{x}_k\vec{x}_l^T\vec{y}_l| < |\vec{x}_k\vec{x}_k^T\vec{y}_k|$$

in spite of the fact that:

$$\vec{x}_k\vec{x}_l^T \not\rightarrow 0$$

At this moment, the nonlinear transformation function F will be effective in attaining the proper recollection results.

For instance, if $\vec{x}_k\vec{x}_l^T$s $(k,l = 1,2,\cdots,s \wedge k \neq l)$ are a constant c, then according to Eq.(2.3), the sign of $\vec{x}_k M$ will coincide with that of $\vec{x}_k\vec{x}_k^T\vec{y}_k$ which indicates that a correct recollection with any one of the key $\vec{x}_k$s will be reached when s is even and the output vectors are specifically coded so that all of them are pairwise different and each of their n components has exactly half the chance to be 1 and -1 respectively to satisfy:

$$\sum_{l=1}^{s}\vec{y}_l = \vec{0} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{2.22}$$

and therefore:

$$\sum_{l\neq k}\vec{y}_l = -\vec{y}_k \tag{2.23}$$

In that case:

$$\begin{aligned}
\vec{x}_k M &= \vec{x}_k \vec{x}_k^T \vec{y}_k + \sum_{l \neq k} \vec{x}_k \vec{x}_l^T \vec{y}_l \\
&= m\vec{y}_k + c\sum_{l \neq k} \vec{y}_l \\
&= m\vec{y}_k + c(-\vec{y}_k) \\
&= (m-c)\vec{y}_k \qquad (2.21) \\
&= \begin{pmatrix} (m-c)y_{k1} \\ (m-c)y_{k2} \\ \vdots \\ (m-c)y_{kn} \end{pmatrix}
\end{aligned}$$

Since $c < m = \vec{x}_k \vec{x}_k^T$, therefore for the sign function *sgn* there exists:

$$\begin{aligned}
\mathrm{sgn}\vec{x}_k M &= \begin{pmatrix} \mathrm{sgn}(\vec{x}_k M)_1 \\ \mathrm{sgn}(\vec{x}_k M)_2 \\ \vdots \\ \mathrm{sgn}(\vec{x}_k M)_n \end{pmatrix} \\
&= \begin{pmatrix} \mathrm{sgn}(m-c)y_{k1} \\ \mathrm{sgn}(m-c)y_{k2} \\ \vdots \\ \mathrm{sgn}(m-c)y_{kn} \end{pmatrix} = \begin{pmatrix} \mathrm{sgn}y_{k1} \\ \mathrm{sgn}y_{k2} \\ \vdots \\ \mathrm{sgn}y_{kn} \end{pmatrix} \qquad (2.25) \\
&= \mathrm{sgn}\vec{y}_k
\end{aligned}$$

In other words, this example suggests that the concentration of their values can be a desired feature of $\vec{x}_k \vec{x}_l^T$'s.

For the reason above, another stochastic characteristic of input vectors *i.e.* the

probability distribution of their components, is of particular concern.

Definition 3 *Assume* $\{\vec{x}_k | k = 1, 2, \cdots, s\}$ *is a set of pre-defined binary vectors. Then the probability distribution of their* i*th component is:*

$$\begin{aligned} P\{x_i = 1\} &= \frac{N_{x_i=1}}{s} \\ P\{x_i = -1\} &= \frac{N_{x_i=-1}}{s} \end{aligned} \qquad i = 1, 2, \cdots, m \tag{2.26}$$

where $N_{x_i=1}$ *and* $N_{x_i=-1}$ *are the number of times when* x_{ki}*s are equal to* 1 *and* -1 *respectively, and* m *is the dimensionality of* $\vec{x}_k$.

Here, x_i functions as a random variable. And in this way, each input pattern can be viewed as an observation of a random vector.

The significance of input vectors' componental probability distribution can be seen as follows. Suppose $\{\vec{x}_k | k = 1, 2, \cdots, s\}$ is a set of bipolar input patterns acquired as a simple random sample of a random vector $\vec{\xi}$, namely, these $\vec{x}_k$s are independent and identically distributed. Also assume that every component of $\vec{\xi}$ is a Bernoulli random variable ξ with one and the same probability distribution, but functions independently:

$$\begin{aligned} P\{\xi = 1\} &= p \\ P\{\xi = -1\} &= 1 - p \end{aligned} \tag{2.27}$$

where p is a constant. Hence each component in an $\vec{x}_k$ has the distribution of this kind, but p is irrelevant to these components:

$$P\{x_{ki} = 1\} = p_i = p = P\{\xi = 1\} \quad , \qquad i = 1, 2, \cdots, m$$

In any specific problem with a given set of data, p_is can be estimated by the componental probability distribution defined in Definition 3.

Notice that:

$$\vec{x}_k \vec{x}_l^T = x_{k1}x_{l1} + x_{k2}x_{l2} + \cdots + x_{km}x_{lm} \tag{2.28}$$

and x_{ki}, x_{li} $(i = 1, 2, \cdots, m)$ are independent, therefore:

$$\begin{aligned} P\{x_{ki}x_{li} = 1\} &= P\{x_{ki} = 1, x_{li} = 1\} + P\{x_{ki} = -1, x_{li} = -1\} \\ &= P\{x_{ki} = 1\}P\{x_{li} = 1\} + P\{x_{ki} = -1\}P\{x_{li} = -1\} \\ &= p^2 + (1-p)^2 \\ P\{x_{ki}x_{li} = -1\} &= P\{x_{ki} = 1, x_{li} = -1\} + P\{x_{ki} = -1, x_{li} = 1\} \\ &= P\{x_{ki} = 1\}P\{x_{li} = -1\} + P\{x_{ki} = -1\}P\{x_{li} = 1\} \\ &= 2p(1-p) \end{aligned} \tag{2.29}$$

So, the expected value of $x_{ki}x_{li}$ is:

$$\begin{aligned} E\{x_{ki}x_{li}\} &= 1 \cdot [p^2 + (1-p)^2] + (-1) \cdot [2p(1-p)] \\ &= (2p-1)^2 \end{aligned} \tag{2.30}$$

On the other hand, if $\vec{x}_k$ and $\vec{x}_l$ are now taken as the specific observations, then $x_{ki}x_{li}$s can be regarded as a series of observations of a random variable $x_k x_l$, the distribution of which has already been described by Eq.(2.29). Therefore, the value of $\vec{x}_k \vec{x}_l^T$ can be estimated in the following manner when m is sufficiently large

$$\vec{x}_k \vec{x}_l^T = mE\{x_k x_l\} \tag{2.31}$$

and by applying Eq.(2.30), there stands:

$$\vec{x}_k \vec{x}_l^T = m(2p-1)^2 \tag{2.32}$$

which is a constant independent of both k and l.

Thus it can be seen that if the componental probability distributions of a set of input vectors $\{\vec{x}_k | k = 1, 2, \cdots, s\}$ are identical, then all the $\vec{x}_k \vec{x}_l^T$s, the factors to introduce crosstalk, tend to be one and the same constant in the statistical sense which may suggest a better self-restraining effect of crosstalk in the associative recall process.

2.4.4 Associative Memory Classifier—No Dynamic Mechanism Be Adopted

When a dynamic mechanism is introduced into associative memory, the process of its information retrieval, either autoassociatively or heteroassociatively, is implemented by continuously feeding the retrieval results of the present iteration back into the memory again and again, until a stable state is reached. Hopfield and Kosko have shown, using Liapunov function, or the energy function of the memory system, that any changes of the associated patterns fed into the memory would cause an energy reduction of the whole system. And since its energy is bounded, a stable state at the local minimum of memory's energy function would be reached finally [112]-[115]. Yet, there is no guarantee that such a retrieval procedure will converge to a final state which is nearest to the initial input pattern.

In the associative recall process for an information recovery problem, both patterns

of an associative pair, most probably distorted, should be provided, and through a series of retrieval feedbacks, the originally stored pair of patterns are supposed to be obtained ultimately. In pattern classification, however, this may not be the case. In the first instance, from the concept point of view, the space of pattern and that of class are different, namely, it is a heteroassociative problem in principle.

Secondly, at the time when an input pattern comes, there is no assumption which class it should belong to, not even an estimation of that. That is to say, there should be no initial pattern of class vector available before retrieval, it can only be obtained from the input vector through the memory. One may find, in dealing with some practical problems, that the real-life input data are usually not quite suitable for associative memory network. Consequently, the class vectors obtained therefrom are seriously distorted even in the first iteration, and a fairly large amount of change in energy has been brought about. This heavily prevents the recollection process from converging to the local minimum caused by the associated pair the input pattern of which is most similar to the associative cue. On the contrary, as the iteration goes on, the pattern changes its form gradually with the system tending to another stable state of energy lower. Therefore, the class vector recalled in the first cycle is the most precise one.

Hence, no dynamic mechanism is adopted in our associative memory classifier (also abbreviated to AMC henceforth). The associative memory used in this thesis for the purpose of pattern classification is the feed forward network illustrated in Figure 2.2.

2.5 Inner Codes Selection—An Example on a Set of Multi-font Chinese Characters

2.5.1 Pattern Data in Use

On constructing an AMC to recognize a set of given patterns, it means to set down all the weight values for each of the network's neurons. At this moment, the input pattern vectors have usually been defined already and are not subject to change any more. The only room left for movement is the selection of inner codes, so that this AMC can reach a better classification performance, which is the basic problem in AMC's construction.

As it has been discussed in the previous section, the choice of inner codes should be based on the characteristics of the input vectors. Therefore, to illustrate some general considerations in inner coding, a certain set of data is needed as an example. In our case, a set of multi-font Chinese characters is employed since the development of associative memory network as pattern classifier is for the potential application to resolve the recognition problem of large number categories, such as that of Chinese characters.

Specifically, the data set being used contains 85 Chinese characters in three different fonts, namely, Song, Kai, and boldface. Hence, there are $85 \times 3 = 255$ different character images in total. Each character image has 56×56 pixels coded bipolarly for the time being. That is to say, the value of a pixel is 1 if the pixel falls on character, and -1 if on background. Finally, the input vector for each character image

is constituted by simply lining the rows or columns of this image matrix up. The input patterns formed as such will also be referred to as original character vectors afterwards.

Obviously, the input vector made up as such might not contain a group of proper features which represent the original pattern in the view of its recognition. Nevertheless, our present attention is concentrated on the construction of AMC by selecting output vectors in accordance with the characteristics of their input ones, and we are designated to establish the system which can tackle practical problems. Therefore this set of input vectors is still appropriate for our present discussion, since their formation does not destroy the fundamental nature of real life data. That is, generally speaking, unlike that of the machine-generated data, their features can not be depicted explicitly in an analytical manner. This in fact also explains the reason for the measurements of the input vectors to be established in previous section on the statistical basis.

2.5.2 Measuring Results on the Pattern Data's Suitability to AMC

The two measurements set up in Section 2.4.3 on the inspection of a certain group of data's suitability to AMC, *i.e.* the average correlation coefficient of this group of vectors and their components' probability distribution, are applied on the set of input patterns of multi-font Chinese characters. The average correlation coefficients of these input vectors which have all the 56×56 components are displayed in the first column of Table 2.1 on the basis of each font of 85 Chinese characters and the entire

set of 255 characters as well. Also, the components' distributions on their probability $p_i = P\{x_i = 1\}$ are illustrated in Figures 2.6 on the same basis, where x_i represents the ith component of the input vector $\vec{x}$. The abscissas p_i are a component's probability $p_i = P\{x_i = 1\}$, and the ordinates $N(p_i)$ represent the number of such components. It is worth mentioning that $N(0)$s, the numbers of components ever-valued as -1 which correspond to the pixels always lying on the background, are not depicted here. Instead, they are listed in the fourth column of Table 2.1. This is merely for the sake of a better display when the differences between $N(0)$ and all the rest $N(p_i)$s $(p_i \neq 0)$ are so large.

Since the average correlation coefficient of a group of vectors may vary only between 0 and 1, and it is expected to approach 0 so as to be fit for an AMC, the feature of this set of input vectors is much less desirable for the μs measured here are all around or above 0.5. Besides, Figure 2.6 shows a much worse situation in the concentration of their components' distribution. As a result, it can be concluded that this set of data is not quite suitable for an associative memory classifier.

2.5.3 Removal of Components with Certainty from Input Vectors

The last column of Table 2.1 shows that in the set of input patterns made up in Section 2.5.1, more than one third of the components $(56 \times 56 \times \frac{1}{3} \approx 1045)$ are always lying on the background. These ever-minus-one-valued components, the probability distribution of which is $p_i = P\{x_i = 1\} = 0$, and the ever-one-valued components, which stand for the pixels always lying on the character and will have the distribution

	I	II	III	IV
Song	0.5651	0.3003	0.2648	1187
Kai	0.6251	0.3471	0.2780	1335
Boldface	0.4641	0.1626	0.3015	1140
Entire Set	0.5115	0.2604	0.2511	1074

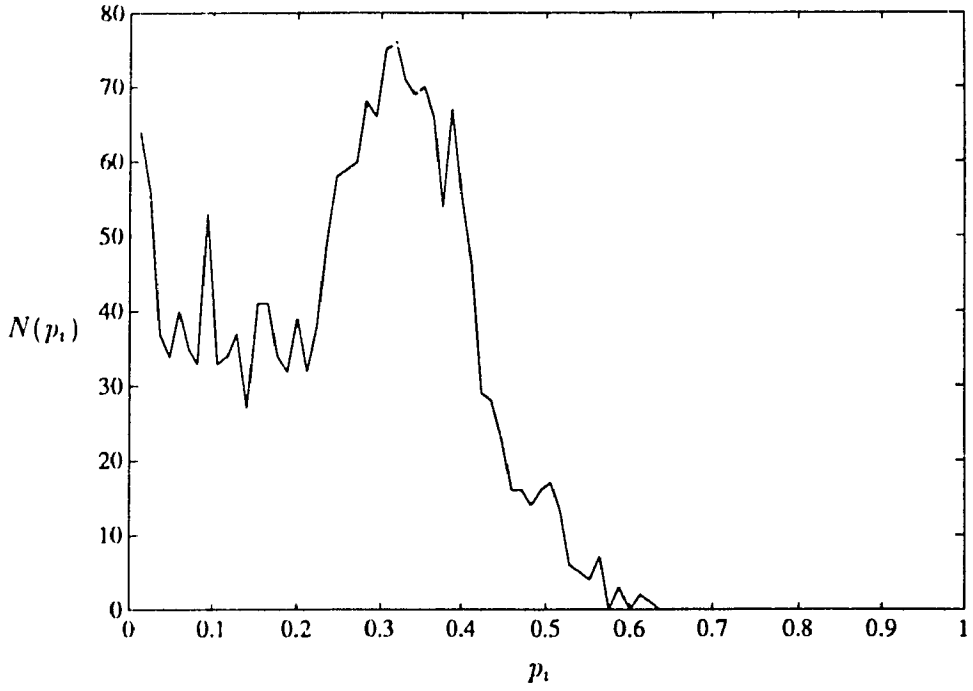
Table 2.1: Some statistics of the character sample spaces.

I. The correlation coefficients of original character vectors in each sample space (*i.e.* those of each font and the entire set of data).

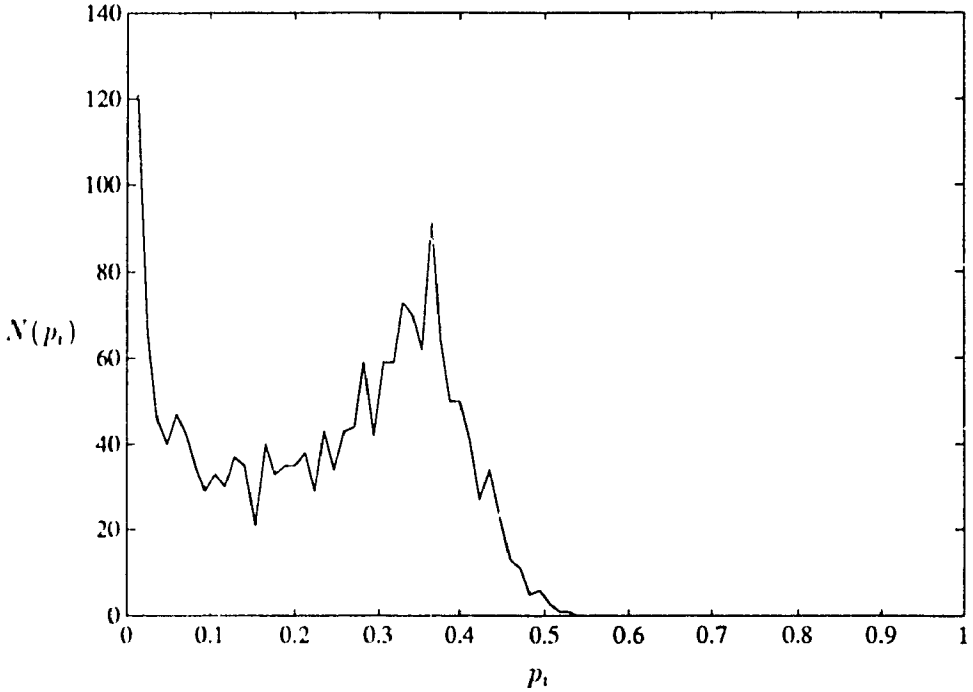
II. The correlation coefficients of original character vectors after the removal of all the always-background components in each sample space.

III. The amounts of reduction between values in column I and II of the same sample space.

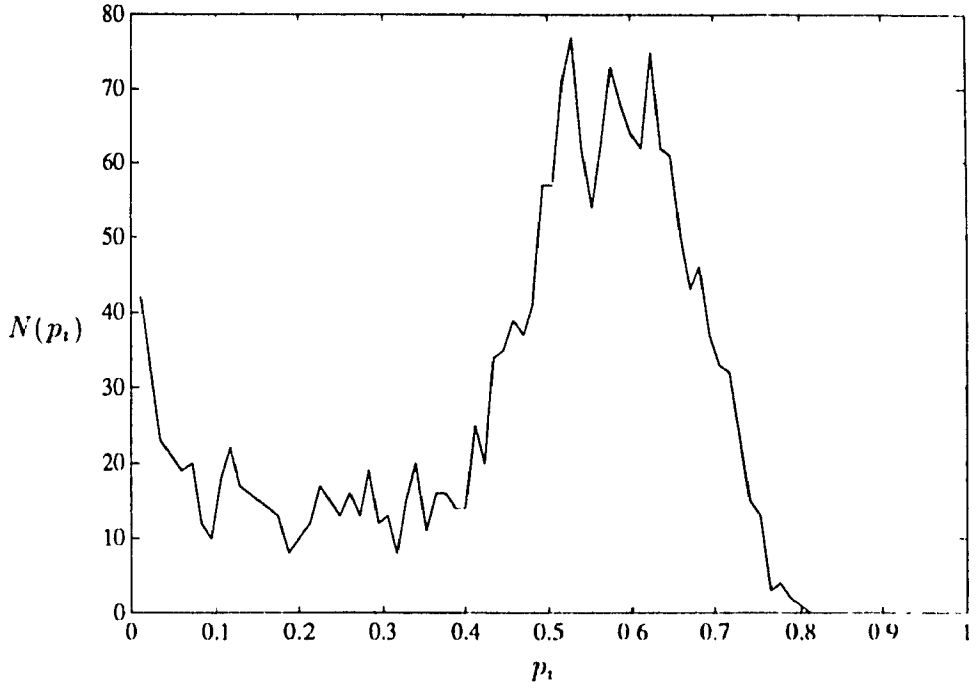
IV. The numbers of the always-background pixels in character images of each sample space.


80
70
60
50
40
30
20
10
0
$N(p_i)$
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
p_i

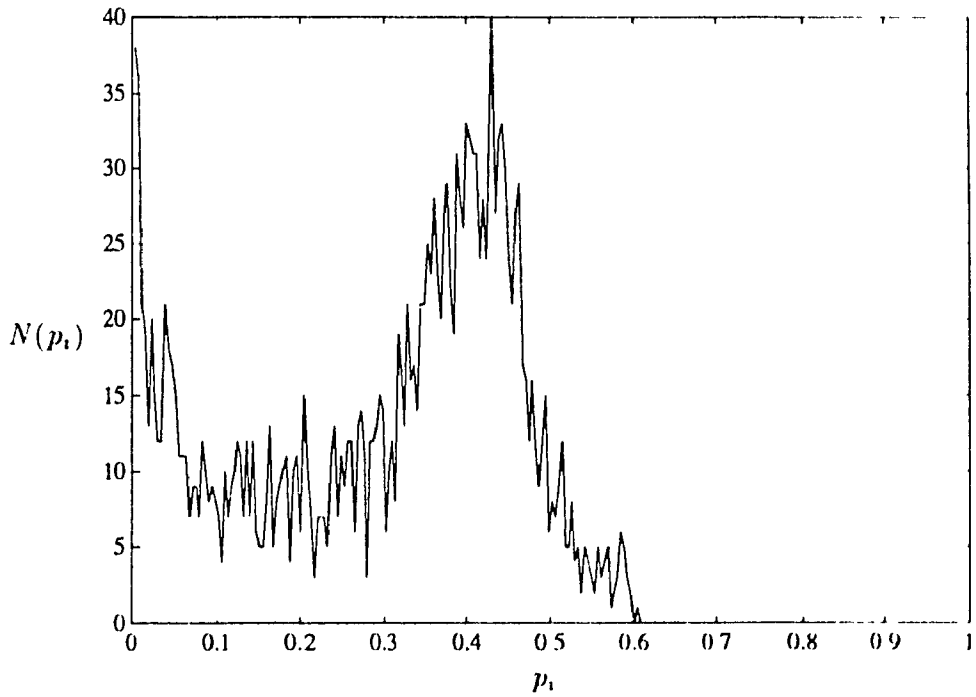
(a)


140
120
100
80
60
40
20
0
$N(p_i)$
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
p_i

(b)



(c)



(d)

Figure 2.6: Components' distributions on their probability $p_i = P\{x_i = 1\}$ of the original character vectors, listed on the basis of each font, *i.e.* Song (a), Kai (b), and boldface (c) respectively, and the entire data set (d)

$p_i = P\{x_i = 1\} = 1$ although actually there is none in our present case, are referred to under a general designation as components with certainty. Obviously, the probability distributions of these components have deteriorated. The components with certainty are addressed particularly here for their unusual effect in associative recall.

Assume $\{\vec{x}_k | k = 1, 2, \cdots, s\}$ is a set of m-dimensional input vectors memorized in M through the corresponding group of output vectors $\{\vec{y}_k\}$. Then during the associative retrieval with key $\vec{x}_k$:

$$\begin{aligned} \vec{x}_k M &= \vec{x}_k \vec{x}_k^T \vec{y}_k + \sum_{l \neq k} \vec{x}_k \vec{x}_l^T \vec{y}_l \\ &= m\vec{y}_k + \sum_{l \neq k} \vec{x}_k \vec{x}_l^T \vec{y}_l \quad , \quad k = 1, 2, \cdots, s \end{aligned} \tag{2.33}$$

Since $|\vec{x}_k \vec{x}_l^T| < m$, for all $l \neq k$, let:

$$\vec{\varsigma}_k = \sum_{l \neq k} \frac{\vec{x}_k \vec{x}_l^T}{m} \vec{y}_l \tag{2.34}$$

the ith component of which is $\varsigma_{ki} = \sum_{l \neq k} \frac{\vec{x}_k \vec{x}_l^T}{m} y_{li}$, therefore:

$$\vec{x}_k M = m\vec{y}_k + m\vec{\varsigma}_k \tag{2.35}$$

Definition 4 *The signal-to-noise ratio of the ith component of $\vec{x}_k M$ is:*

$$SNR(\vec{x}_k M)_i = \frac{m y_{ki}}{m \varsigma_{ki}} = \frac{y_{ki}}{\varsigma_{ki}} \tag{2.36}$$

Hence, the following understanding on the impact of components with certainty to associative recollection can be reached:

Theorem 1 *Suppose $\{\underline{\vec{x}}_k | k = 1, 2, \cdots, s\}$ is a set of input vectors, each of which is obtained by augmenting the corresponding $\vec{x}_k$ with $\underline{m}$ additional components with*

certainty, and the associative memory formed by these s pairs of $\underline{\vec{x}}_k$ *and* $\vec{y}_k$ *is* $\underline{M}$. *Then, we have:*

$$\lim_{m \to \infty} SNR(\underline{\vec{x}}_k \underline{M})_r = 0 \tag{2.37}$$

Proof:

Without loss of generality, it can be presumed that these components with certainty are all minus-one-valued, and concatenated to the end of $\vec{x}_k$, that is:

$$\begin{aligned} \underline{\vec{x}}_k &= (x_{k1}, \cdots, x_{km}, \underbrace{-1, \cdots, -1}_{\underline{m}}) \\ &= (\vec{x}_k, \underbrace{-1, \cdots, -1}_{\underline{m}}) \end{aligned}$$

Hence:

$$\begin{aligned} \underline{\vec{x}}_k \underline{\vec{x}}_k{}^T &= (\vec{x}_k, -1, \cdots, -1) \begin{pmatrix} \vec{x}_k^T \\ -1 \\ \vdots \\ -1 \end{pmatrix} \\ &= \vec{x}_k \vec{x}_k^T + \underline{m} \end{aligned}$$

Now that in the information recollection stage:

$$\begin{aligned} \underline{\vec{x}}_k \underline{M} &= \underline{\vec{x}}_k \sum_{l=1}^{s} \underline{\vec{x}}_l{}^T \vec{y}_l = \sum_{l=1}^{s} \underline{\vec{x}}_k \underline{\vec{x}}_l{}^T \vec{y}_l \\ &= \sum_{l=1}^{s} (\vec{x}_k \vec{x}_l^T + \underline{m}) \vec{y}_l \\ &= \sum_{l=1}^{s} \vec{x}_k \vec{x}_l^T \vec{y}_l + \sum_{l=1}^{s} \underline{m} \vec{y}_l \\ &= \vec{x}_k \vec{x}_k^T \vec{y}_k + \sum_{l \neq k} \vec{x}_k \vec{x}_l^T \vec{y}_l + \underline{m} \sum_{l=1}^{s} \vec{y}_l \end{aligned} \tag{2.38}$$

Let the summation of all the output vectors be $\vec{\varsigma}$:

$$\vec{\varsigma} = \sum_{l=1}^{s} \vec{y}_l \tag{2.39}$$

So:

$$\vec{x}_k \underline{M} = m\vec{y}_k + m\vec{\varsigma}_k + \underline{m}\vec{\varsigma} \tag{2.40}$$

According to the above definition, for the ith component of $\vec{x}_k \underline{M}$:

$$\begin{aligned} \mathrm{SNR}(\vec{x}_k \underline{M})_i &= \frac{m y_{k_i}}{m\varsigma_{k_i} + \underline{m}\varsigma_i} \\ &= \frac{y_{k_i}}{\varsigma_{k_i} + (\underline{m}/m)\varsigma_i} \end{aligned} \tag{2.41}$$

Since the value of m is fixed, the theorem is tenable.

A comparison between Eq.(2.36) and Eq.(2.41) indicates clearly that the introduction of components with certainty are absolutely useless in differentiating different patterns other than causing more crosstalk. This can also be seen from Eq.(2.40). With $\underline{m}$ getting larger and larger, the effect of crosstalk becomes more and more severe, until finally the recollection results will all be dominated by the summation of the whole set of output vectors regardless of the applied associative cues. In a word, any component of this sort carries no information concerning the input vector's identification. Going a step further, it is reasonable to believe that actually the amounts of information taken along by the components other than these are also different, and the most discriminative ones should be those with $p_i = \frac{1}{2}$.

For this reason, all the components with certainty which in fact come solely from the pixels lying along the four edges of the character image are eliminated from the corresponding input pattern, and the input vectors formed hereof will be used in

our future studies. The average correlation coefficients of each group of character patterns thus generated are shown in column II of Tabel 2.1. In all these cases, there are reductions in the corresponding average correlation coefficients and the amounts of these reductions are listed in column III. The largest decrease in that of boldface characters implies that the correlation of their input patterns is affected more by these pixels since their strokes are thicker. Characters of font Kai gain more improvement than those of font Song because the sizes of the former are smaller, and therefore more such pixels are eliminated. When these three fonts are put together, there are less always-background pixels to be removed. However, the correlation of all the 255 characters is still smaller than those when only 85 characters of font Song or Kai are considered.

Incidentally, it is worth mentioning as well that no specific preprocessing measures have ever been taken in our current data, especially that of normalization. The original character size of font Kai is the smallest, which means its characters are composed of less pixels, while the character strokes of font boldface are the thickest among the three. For this reason, the statistical results of the former font is expected to be the worst while that of the latter the best. These have already been manifested in both Table 2.1 and Figure 2.6.

2.5.4 Selection of Inner Codes

It has been indicated that a pattern classification process may be viewed as an associative mapping from the pattern space to the category one. Evidently, these are

two different sorts of signals, and simply for this reason, the mapping is not taken as an autoassociative one in concept. However, this logical difference does not prevent a pattern and its category signal bear a common vector meaningful in each of their own data space. And this will lead to an autoassociative mapping in practice.

Generally speaking, in the selection of inner codes, any schemes could be utilized in regard to information memorizing. The most intuitive way, for instance, is to use an input vector itself as its output pattern as well. However, whether a specific choice is an appropriate one will, in the final analysis, be conditioned by the classification potential of the system. Now, is there any possibility for an AMC to be an autoassociative network?

Back to our discussion in Section 2.4.3, Eq.(2.32) shows that $x_k^i x_l^i \to 0$ when $p = \frac{1}{2}$. This means that, in this case, there tends to be no crosstalk being introduced. Hence, the input patterns themselves can be taken as their associated output data, which turns the AMC into an autoassociative network, as well as any other inner coding schemes that may keep the distinctiveness of the original inputs. Actually, under this circumstance, there will be $p = \frac{1}{2}$ for the components of output vectors. That is to say, each same component in $\vec{y}_l$ $(l = 1, 2, \cdots, s)$ will have half the chance to be positive (1) and the rest negative (-1). Therefore, statistically, there is a tendency for the crosstalk of $\vec{y}_l$s $(l = 1, 2, \cdots, s \quad l \neq k)$, if there is any, to be counterbalanced from one another, as can be seen from Eq.(2.3).

Unfortunately, the results of our statistics in Figure 2.6 manifest the biases of components' distribution to the lower $p_i = P\{x_i = 1\}$ side, instead of concentration

near $p_i = 0.5$. Namely, they take the value -1 more likely which corresponds to a preponderance phenomenon of background pixels in our character images. In short, the above condition is not satisfied here. And for this reason, the input vectors are not suitable to be the inner codes of themselves in the present problem. Thus it can be seen that, in solving a real-life problem, the possibility to use autoassociative memory may be ruled out for the practical rather than logical reasons.

Then what kind of vector is suitable for associating with the input vectors of this group of Chinese character images? As it has been mentioned in Section 2.4.2, if input vectors are perpendicular to one another, then a perfect recall could be reached. Although this is impossible in our present case, a group of orthogonal vectors could be chosen as inner codes, since a recalled vector obtained through associative recollection and distorted most probably is nothing but one in the space spanned by all the associated inner code vectors. The farther these base vectors fall away, the less it is possible for them to interfere with one another in the retrieval process, and the best way to achieve that goal is simply having these vectors orthogonal to one another.

In our problem at hand, the row vectors in Hadamard transformation matrix, called Hadamard vectors, are chosen. Hadamard transformation matrices can be generated in a simple recursive way [116], the amount of vectors available then is always in a number of the power of 2. Such a matrix with its dimensionality being 8

is as follows:

$$H_8 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}$$

Obviously, vectors obtained as such are orthonogal, and all their corresponding components except the first one have exactly half the chance being both positive and negative. As a result, the performance of the whole memory is expected to be better when all of these vectors are used for association. This also explains why 85 Chinese characters of three fonts are involved in our data set when $2^8 = 256$ Hadamard vectors are under consideration.

2.6 Simulation on AMC for Character Recognition

85 real-life Chinese characters were collected in three different fonts. These characters are illustrated in Appendix A. They are employed in the previous section as a set of data obtained to discuss the inner coding problem. This group of characters is simply the first 85 in our data base of the most commonly used 3114 printed characters each in three different fonts, and Hadamard vectors have been chosen as the set of

output vectors for them. Obviously, the applicability of associative memory as pattern classifier is decided ultimately by its recognition performance.

2.6.1 Simulation Network and Its Recognition Criteria

The establishment of the associative memory classifier is the process described in Eq.(2.2) of Section 2.3.2, with the associated pairs composed of $85 \times 3 = 255$ input vectors formed pixel by pixel from the non-always-background part of the character images, and 255 Hadamard vectors with dimensionality of 256.

In the recognition phase, classification is made based on the output vector recalled through the memory. Due to the disturbance of the contamination on the inputs to AMC and the crosstalk during the associative retrieval, the recollected outcomes are also contaminated rather than the pure Hadamard vector themselves. Actually, any one of the recollected vector is a linear combination of these bases. So it needs to be projected thereafter to all the Hadamard vectors to estimate its closenesses with each of them.

Definition 5 *Suppose $\vec{y}$ is the recalled vector, its similarity value S_k with an output vector, e.g. a Hadamard vector in the present case, $\vec{y}_k$ is:*

$$S_k = \frac{n + \sum_{i=1}^{n} y_i \oplus y_{k,i}}{2} \tag{2.42}$$

where y_i and $y_{k,i}$ are the ith components of vectors $\vec{y}$ and $\vec{y}_k$ respectively, $\oplus$ is the exclusive-nor operation, and n is the dimensionality of output vectors $\vec{y}_k$s (which equals 256 currently).

Also, we define the largest similarity value of a set of S_k as follows.

Definition 6 *The largest similarity value (LSV) of a set of* S_k ($k = 1, 2, \cdots, s$) *is* S_l *which satisfies:*

$$S_l \geq S_k \quad , \qquad \forall\, k \neq l,\ k = 1, 2, \cdots, s$$

Then the recognition can be conducted based on the following criteria:

Criterion 1 *Classification Rule:*

An input pattern is classified as character k, *if for the set of similarity values acquired in the above way, there is:*

$$S_k > S_l \quad , \qquad \forall\, l \neq k,\ l \in \Psi \quad \wedge \quad \Psi = \{1, 2, \cdots, 255\}$$

where 255 *is the number of currently used output vectors.*

That is to say, the input pattern is classified as character k, if the retrieved vector $\vec{y}$ has the solely largest similarity value S_k with Hadamard vector $\vec{y}_k$.

Criterion 2 *Rejection Rule:*

An input pattern is rejected, if for the set of similarity values,

$$S_k > S_l \quad , \qquad \forall\, l \neq k,\ k \in \psi,\ l \in \Psi$$
$$\wedge \quad \psi \subseteq \Psi,\ |\psi| \geq 2,\ \ \Psi = \{1, 2, \cdots, 255\}$$

that is, the number of elements in set ψ *is more than one.*

This means that an input pattern is rejected whenever its recalled vector has LSV with more than one output vector.

Criterion 3 *Correctness Rule:*

An input pattern is said to be recognized correctly if it is the pattern of character k *in a certain font, even though being distorted in a way or another, and has been classified as this character of the same font.*

The above three creteria are in fact unnecessarily strict in the view of character recognition for under a practical situation of this kind our interest is only concentrated on the proper identification of a character itself, its printing style is rarely concerned. Nonetheless, these recognition criteria contradict with this common knowledge in the following two cases which may well occur in multi-font character recognition. First of all, our Correctness Rule will treat a character pattern which has been classified into one of its other fonts as misclassification. Next, even if the largest similarity value of the recollection outcome is shared among the output vectors associated with the inputs of one and the same character in different fonts include the right character image, such a result is still going to be rejected. Both of these, however, should be acceptable in terms of character recognition.

Then why are these criteria adopted? For the time being, we are in the stage of exploring the possibility of employing associative memory network as pattern classifier rather than developing a real AMC for a specific application. It is hoped that this novel neural classifier could be built on a robust basis, and the multi-font data is used

to introduce intentionally some similarity among character images. Besides, this new sort of pattern classifier is aimed at solving recognition problems involving a large number of categories, that is, instead of several or a few dozens, it is expected to deal with hundreds or even thousands of categories. Although this has already been fairly large, the categories of Chinese characters are counted in tens of thousands. However, a problem of such a magnitude is usually not attempted to be tackled with a classifier of a single stage. The AMC being developed currently may be used in the future as parts of a whole network to resolve it. This, again, offers another reason to apply tough criteria on the network being studied at the present time.

2.6.2 Testing Data Groups

Six groups of data are used in testing the recollection performance of this associative memory classifier, each contains all the 255 characters. The first group is the original data used in the memory establishment. In group two to four, there are ten, twenty, and forty percent of random noise pixels in the non-always-background area of character images respectively. Those pixels influenced by noise are changed into their opposite states, that is, a 'character' pixel will become a 'background' one leaving a 'hole' on the stroke it lies, and *vice versa*. A four pixel wide horizontal 'background' stripe covers the middle part of every character image in data group five, while in the last group a 'character' stripe of similar nature is applied. These brute force distortions are designed merely for the test of AMC's adaptability to noise with consistency and regularity. Some character images of these data groups are shown in Figure 2.7.

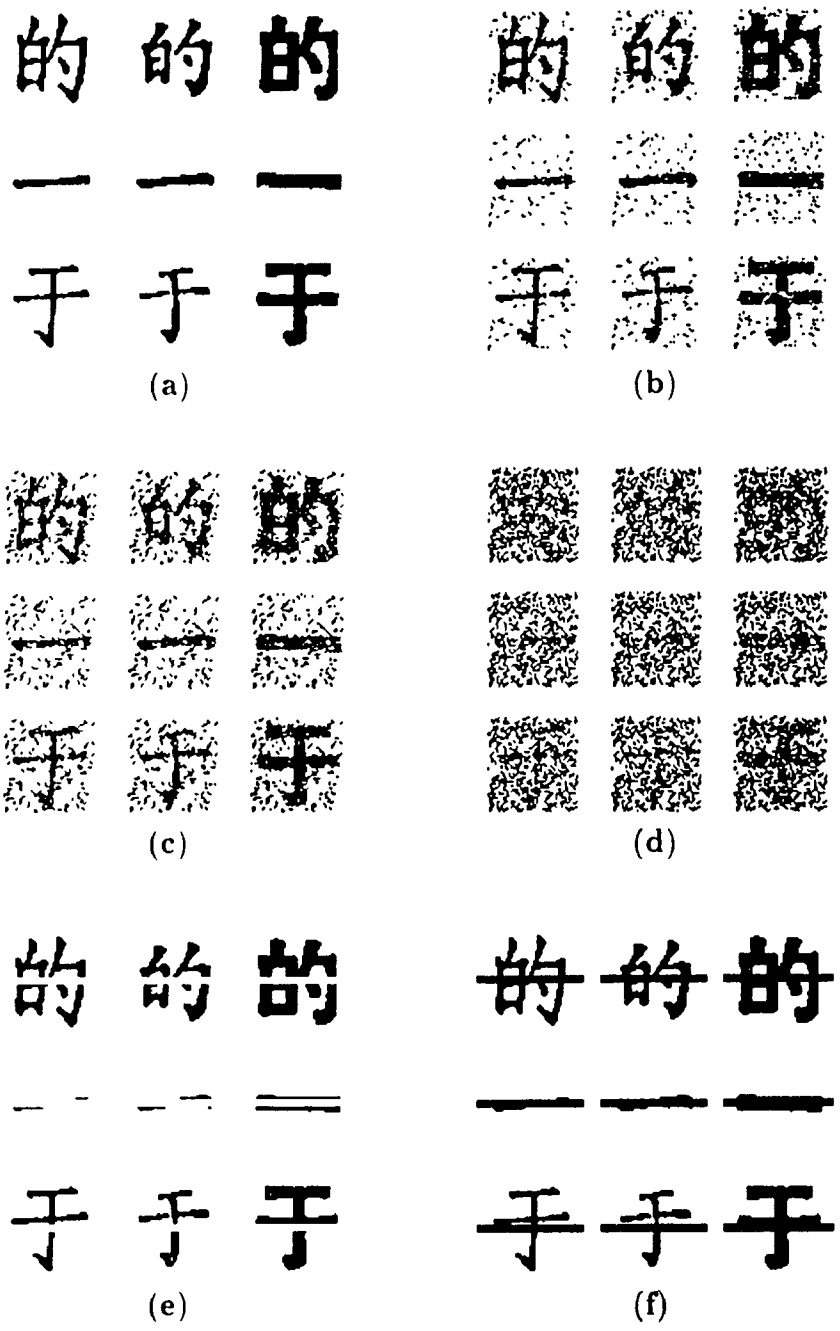


Figure 2.7: Some character images of the testing data groups I (a), II (b), III (c), IV (d), V (e), and VI (f).

2.6.3 Results and Analyses

The results of classification in each case are displayed in Table 2.2. The numbers of misclassification (above) and rejection (below) are listed first on the basis of each font, and then added up to show the whole picture of the entire data group, together with the corresponding recognition, rejection, and error rates of the latter. Table 2.2 demonstrates the effectiveness of applying associative memory to recognize this set of data made from the multi-font Chinese character images. Although these original character vectors are not quite suitable for associative retrieval as has been discussed in the previous section, high recognition rates all above 98% are still achieved, even in the presence of severe random noise (up to 20% in our experiment). In the testing data group four to six, when input patterns are exceedingly distorted, the recognition results, with their rates all above 93%, are still encouraging.

Since the classification is achieved based on LSVs, they show the performance of the memory system in more detail. This is because at the stage of information retrieval, the larger the LSV is, the closer the recalled vector gets to a specific inner code. The exact form of an inner code vector is recollected if LSV achieves its maximum, the dimensionality of the Hadamard vectors used, and if this is true for an average LSV, it means that such a group of patterns can be retrieved without any distortion. In this sense, LSV can be regarded as an indication of the confidence of this classification process, and therefore, it is a major measurement on the performance of an associative memory classifier.

Table 2.3 presents the maximal, minimal, and average values of LSV among each

	Testing Data Group					
	I	II	III	IV	V	VI
Song	0	3	1	1	2	4
	0	0	0	2	0	1
Kai	1	1	2	6	3	5
	0	1	0	5	2	5
Boldface	1	0	1	2	1	1
	0	0	0	1	0	0
Entire Set	2	4	4	9	6	10
	0	1	0	8	2	6
Recognition Rate	99.22%	98.04%	98.43%	93.33%	96.86%	93.73%
Rejection Rate	0.0%	0.39%	0.0%	3.14%	0.78%	2.35%
Error Rate	0.78%	1.57%	1.57%	3.53%	2.35%	3.92%

Table 2.2: The numbers of misclassification and rejection in different testing data groups, listed in line with each font and the entire data set, together with the corresponding recognition, rejection, and error rates of the latter.

kind of font and the whole data set on the basis of every testing data group. It is very interesting to notice that the maximal LSV's of the whole data samples come from those of font boldface, and the minima from those of font Kai in all the six testing data groups. Besides, the average values of the former font are always the largest while those of the latter font the smallest, and also always the averages over all three fonts are larger than the averages of font Song. These mean that in associative recollection, the results of boldface characters are the best, and the results of Song characters are better than those of Kai's. This conclusion coincides with the data analysis made in the previous section. However, it should be noted that the above discussion is based on an average point of view, it is just a description of the general tendency, and therefore something unusual may happen occasionally such as the misclassification of a boldface character in the first data group.

The approach of pattern classification using associative memory has intrinsically a strong random noise resistance nature, because each of the correct recollection is achieved only after the survival of heavy interference of crosstalk by virtue of their self-restraining ability acquired through appropriate inner coding and the nonlinear transformation mechanism of the network. In such severe cases of our experiment as ten or twenty percent of random dots are in existence, Table 2.3 shows the average LSVs do not change much, and correspondingly rather slight deteriorations in recognition rate have been observed (see Tabel 2.2). Moreover, even when forty percent random noise pixels made character images indiscriminate to the human visual system, they still can be classified by the associative memory network with a fairly high

		Testing Data Group					
		I	II	III	IV	V	VI
Song	maximum	196	194	195	181	189	193
	minimum	153	152	154	152	149	152
	average	175.07	174.64	174.54	167.26	168.27	172.09
Kai	maximum	193	187	184	177	178	181
	minimum	149	148	148	147	148	151
	average	169.46	169.12	169.13	161.54	163.45	165.07
Boldface	maximum	204	205	202	189	197	203
	minimum	163	162	164	153	149	161
	average	187.74	187.65	186.47	175.27	181.32	183.48
Entire Set	maximum	204	205	202	189	197	203
	minimum	149	148	148	147	148	151
	average	177.42	177.13	176.71	168.02	171.01	173.55

Table 2.3: The statistical data of largest similarity value (LSV) during the information retrieval process of different testing data groups. The results are listed on the basis of each font and the entire sample set. The range of LSV varies from 0 to 256.

recognition rate.

Due to the fact that the input vectors currently used are formed pixel by pixel from the original character images, data groups five and six may possess some other special meanings. In the view of associative memory, the correct information retrieval from such forms of data represent the memory's ability of recollection on the basis of partial information, which is one of the associated retrieval apabilities inherently possessed by human beings. In the present case of character recognition, such distortions manifest the potential of the system to be free from errors caused by breaks of a stroke due to the wearing of type, or connection among strokes caused by heavy ink *etc.* in real printing.

In the design of a practical pattern classifier, some remedial measures may be taken when a rejection occurs. Comparatively speaking, misclassification is more troublesome since the system itself can by no means identify it. In the above experiments, except those misclassifications caused by the Correctness Rule, the rest comes mainly from three sources respectively. First of all, about 60% of these misclassifications in testing data groups one to five are caused by confusions among the following characters:

了 不 上 个 于 下 子

which are obviously similar to one another to some extent in their image structures (both of character and background). Next, most other errors in these testing data groups are due to severe distortion of the input patterns in the ways of groups four

and five. Thirdly, the confusions with character " — ", if not those among the above seven characters, account for the major errors occurred in the last group. This sheds some light that the associative memory may have the ability to extract some similarities among the input pattern vectors. In the present case, since the original character vectors are constituted on the pixel basis of the character images, some structural resemblances have been detected. This happened in the presence of many other character patterns quite different from these. Therefore, the categories of those misclassified or even rejected characters can be merged, and further classification mechanism can be introduced to deal with such smaller sets of data.

2.7 Discussion

Our investigation in this chapter has discovered the applicability of associative memory as a pattern classifier. Pattern classification, in the logical sense, can be regarded as associative mapping. However, in dealing with a practical problem, an associative memory classifier will function properly only after a careful consideration has been given to the selection of output vectors, or inner codes, which are associated with the patterns to be recognized. Also, the choice of these inner codes should be based on some intrinsic characteristics of their associated input pattern data.

For the same set of input patterns, the performances of AMCs formed with different sets of inner codes may be quite distinct. For instance, with the group of multi-font Chinese character data currently in use, a failure will occur if the network is made to be an autoassociative one. Nevertheless, our experiment shows that if

Hadamard vectors are chosen as the inner codes, a rather good recognition behaviour can be observed. This may bring forth a series of other questions. If different inner coding schemes may lead to different recognition performance of an AMC, then does there exist any one optimal? And if so, how can a better scheme, or even an optimal one, be obtained? The next chapter will be devoted to answer some of these questions.

Chapter 3

Inner Coding Schemes in Associative Memory Classifier

3.1 Introduction

It has been found in the previous chapter that an associative memory can be used as a pattern classifier to recognize fairly large categories of patterns. Nevertheless, for an associative memory to function properly, a serious consideration on the selection of inner codes associated with the patterns to be recognized is a necessity. Different sets of inner codes may result in totally different classsification performances. This captures our interest in seeking better, or even optimal, inner coding schemes. To do so, it is necessary to set up an efficient tool to evaluate the recollected outcomes of an associative memory. This will be our major objective described in Section 3.2.

Before any attempt is made in finding the optimal coding schemes, their existence should be guaranteed first. This will be proved in Section 3.3 based on the limited

number of possible coding schemes with output vectors of finite dimensionality. Although it is very powerful in theoretical studies, this same limited nature will also rule out the possibility of utilizing enumeration as a practical way to find the optimum, for this may turn out to be a non-polynomial problem.

In Chapter 2, Hadamard vectors have been chosen as a set of favourable inner codes in light of the analyses on the characteristics of the input data. But, simply selecting a set of output vectors as a whole does not fully accomplish the task of inner coding, since it still remains open at this moment on which output vector should be used to associate with a particular input pattern. In our previous experiment, this was done merely according to the natural orders of data storing in both the input pattern and Hadamard vector sets without awareness of such a problem explicitly. However, this helps to suggest a practical way to carry out inner coding in a two step manner. The first is the selection of a set of vectors which take desired forms or have some specified features. The second is the association of each input pattern with a specific vector of the inner code set. This second step will be called a pairing process, and the way to associate each output vector with an input pattern referred to as a pairing scheme.

An optimal pairing scheme will be addressed in Section 3.4 under some ideal conditions when Hadamard vectors are chosen to be the inner codes. The strategy to apply it to practical data for the better pairing schemes will also be discussed. Its effectiveness will be demonstrated in Section 3.5 with the same set of real-life character data as has been employed in the previous chapter.

3.2 Evaluation of Associative Recollection Performance

3.2.1 Root Mean Square Error of the Associative Recollection Process

Different sets of inner codes may result in different recognition behaviour of an associative memory classifier. Therefore, attaining a better system performance through wise inner coding is of great interest. However, before any systematic studies on this are to be conducted, an analytical way to estimate the errors during the associative recollection has to be set up first as an effective measurement of an AMC's behaviour. To this end, more details about the information retrieval process of associative memory should be looked into.

Suppose $\{(\vec{x}_k, \vec{y}_k)|k = 1, 2, \cdots, s\}$ are a group of associated pattern pairs, where $\vec{x}_k = (x_{k1}, x_{k2}, \cdots, x_{km})$, $\vec{y}_k = (y_{k1}, y_{k2}, \cdots, y_{kn})$ are bipolar row vectors in m- and n-dimensional spaces respectively, *i.e.* all their components may take the value of either 1 or -1. Then the associative memory M, where all these s pattern pairs are stored, is:

$$M = \sum_{k=1}^{s} \vec{x}_k^T \vec{y}_k \tag{3.1}$$

In the process of information recollection, the recalled output $\vec{y}$ is some nonlinear transformation F of the product of the associative cue $\vec{x}$ and the memory matrix M, also called retrieval or recollection product:

$$\vec{y} = F(\vec{x}M) \tag{3.2}$$

where in this thesis F takes the form of function depicted in Eq.(2.7) unless specified

otherwise.

Assume, at present, the input pattern $\vec{x} = \vec{x}_k$, therefore the retrieval product $\vec{x}_k M$ is:

$$
\begin{aligned}
\vec{x}_k M &= \vec{x}_k \sum_{l=1}^{s} \vec{x}_l^T \vec{y}_l = \sum_{l=1}^{s} \vec{x}_k \vec{x}_l^T \vec{y}_l \\
&= \sum_{l=1}^{s} c_{kl} \vec{y}_l = \sum_{l=1}^{s} c_{kl} \begin{pmatrix} y_{l1} \\ y_{l2} \\ \vdots \\ y_{ln} \end{pmatrix}^T \\
&= c_{k1} \begin{pmatrix} y_{11} \\ y_{12} \\ \vdots \\ y_{1n} \end{pmatrix}^T + c_{k2} \begin{pmatrix} y_{21} \\ y_{22} \\ \vdots \\ y_{2n} \end{pmatrix}^T + \cdots + c_{ks} \begin{pmatrix} y_{s1} \\ y_{s2} \\ \vdots \\ y_{sn} \end{pmatrix}^T \\
&= \begin{pmatrix} c_{k1}y_{11} + c_{k2}y_{21} + \cdots + c_{ks}y_{s1} \\ c_{k1}y_{12} + c_{k2}y_{22} + \cdots + c_{ks}y_{s2} \\ \vdots \\ c_{k1}y_{1n} + c_{k2}y_{2n} + \cdots + c_{ks}y_{sn} \end{pmatrix}^T \\
&= \begin{pmatrix} z_{k1} \\ z_{k2} \\ \vdots \\ z_{kn} \end{pmatrix}^T \\
&= \vec{z}_k
\end{aligned}
\tag{3.3}
$$

Let:

$$C = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1s} \\ c_{21} & c_{22} & \cdots & c_{2s} \\ \vdots & & & \\ c_{s1} & c_{s2} & \cdots & c_{ss} \end{bmatrix} \tag{3.4}$$

where

$$c_{kl} = \vec{x}_k \vec{x}_l^T \qquad , \qquad k, l = 1, 2, \cdots, s \tag{3.5}$$

is the inner product of pattern $\vec{x}_k$ and $\vec{x}_l$, and hence called the correlation number of these two patterns. Therefore, C is referred to as the system parameter matrix since each of its element is the correlation number of a corresponding pair of input patterns memorized in this associative memory network. Evidently, C is symmetric for:

$$c_{kl} = \vec{x}_k \vec{x}_l^T = \vec{x}_l \vec{x}_k^T = c_{lk}$$

Also let:

$$Y = \begin{bmatrix} \vec{y}_1 \\ \vec{y}_2 \\ \vdots \\ \vec{y}_s \end{bmatrix} = \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1n} \\ y_{21} & y_{22} & \cdots & y_{2n} \\ \vdots & & & \\ y_{s1} & y_{s2} & \cdots & y_{sn} \end{bmatrix} \tag{3.6}$$

$$Z = \begin{bmatrix} \vec{z}_1 \\ \vec{z}_2 \\ \vdots \\ \vec{z}_s \end{bmatrix} = \begin{bmatrix} z_{11} & z_{12} & \cdots & z_{1n} \\ z_{21} & z_{22} & \cdots & z_{2n} \\ \vdots & & & \\ z_{s1} & z_{s2} & \cdots & z_{sn} \end{bmatrix} \tag{3.7}$$

here Y is called a code matrix, and Z the retrieval or recollection product matrix.

Hence, Eq.(3.3) can be written as below:

$$\vec{z}_k = C_k Y \tag{3.8}$$

where C_k is the kth row of matrix C. And the above defined matrices have the following relation:

$$Z = CY \tag{3.9}$$

The importance of describing the information retrieval process on the entire vector set base by means of matrices can become clear in the discussions later on.

Now, apply nonlinear transformation F on $\vec{z}_k$, or the row vectors of Z. This attains:

$$\begin{aligned} \hat{\vec{y}}_k &= (\hat{y_{k1}}, \hat{y_{k2}}, \cdots, \hat{y_{kn}}) \\ &= F(\vec{z}_k) \qquad , \quad k = 1, 2, \cdots, s \end{aligned} \tag{3.10}$$

So according to Eq(2.7):

$$\hat{y_{kl}} = \begin{cases} 1, & \text{if } z_{kl} > 0 \\ -1, & \text{if } z_{kl} < 0 \end{cases} \qquad l = 1, 2, \cdots, n \tag{3.11}$$

Therefore, the associative recollection error of the memory system can be estimated as follows:

Definition 7 *Presume $\{\vec{x}_k | k = 1, 2, \cdots, s\}$ is a set input patterns stored in memory M by associating with inner codes $\{\vec{y}_k\}$ separately, and $\hat{\vec{y}}_k$ $(k = 1, 2, \cdots, s)$ is the recollected vector under the application of key $\vec{x}_k$, that is:*

$$\hat{\vec{y}}_k = F(\vec{x}_k M) \quad , \qquad k = 1, 2, \cdots, s$$

Then the root mean square (RMS) error of associative recollection under the inspecting data set $\{\vec{x}_k\}$ *is:*

$$\varepsilon_{(RMS)}(\{\vec{y}_k\},\{\vec{x}_k\}) = \left[\frac{1}{s}\sum_{k=1}^{s} \| (\vec{y}_k - \hat{\vec{y}}_k) \|^2\right]^{\frac{1}{2}} \tag{3.12}$$

In Eq.(3.12), $\vec{y}_k$ is the expected response, or the original form of the inner code, and $\hat{\vec{y}}_k$ is the actual response of the network. The smaller the value of ε is, the better the inner coding scheme the memory system has. Therefore, Eq.(3.12) can serve as a judgement of different inner coding schemes.

3.2.2 Optimum Criterion

Eq.(3.12) is an intuitive way and also a very useful tool to evaluate the appropriateness of a certain set of inner codes for an associative memory designed to recognize a specific group of patterns. It has several interesting aspects which deserve to address before the criterion for optimal inner code set can be laid down.

First and foremost, according to Eq.(3.12), ε is a function of not only the inner code set $\{\vec{y}_k\}$ but also the inspecting data set $\{\vec{x}_k\}$. Therefore, in order to institute comparisons among different coding schemes, a standand set of inspecting data should be set up, based on which the associative recollection error of each inner coding scheme is calculated. In Definition 7, the pattern vectors memorized in AMC is employed as an example. This is absolutely not a necessity. For instance, if the AMC will be working under an environment in which the input patterns may frequently suffer from some sorts of distortions, then introducing these distortions into the inspecting data

during the selection of the AMC's inner codes will be more preferable.

Next, ε is calculated on the distances between all pairs of vectors $\vec{y}_k$ and $\hat{\vec{y}}_k$. When $\hat{\vec{y}}_k = \vec{y}_k$, ε reaches its minimum 0. However, here associative memory is used for classification purpose. A correct identification can be achieved when $\hat{\vec{y}}_k$ is closer to $\vec{y}_k$ than any other inner codes, instead of just being $\vec{y}_k$. This can also be seen from the simulation results presented in the last chapter. By checking the recognition rate and different LSV values of every testing data group, it seems almost certain when each LSV reaches its average value of the whole group, a correct classification is attained. In all these cases, it is merely 170 or so out of 256. Besides, it is found in our experiments that if the average LSV starts to decline from the points around this, the recognition rate will drop rather rapidly. Now, notice that the similarity value and the difference or distance between the recollected vector and inner codes are actually the complementary computation, and so are recognition rate and recollection error in a sense, the following conclusion is made:

Conclusion 2 *The associative recollection error varies monotonically with the distances between each pair of the expected responses and the actually retrieved vectors, and a proper nonlinear relation may reflect more accurately its meaning in the classification sense.*

Of course, the specific form of this nonlinear function may be application dependent.

Thirdly, if in a practical situation some patterns appear more often than others, it is realistic then to factor the distance of each vector pair with the appearance

frequency value of the corresponding pattern instead of just taking the average as in the RMS error. In fact, what is considered in the latter is simply the case of uniform appearance frequency.

Finally, to make this estimation on the AMC's performance meaningful, the trivial cases should be excluded. These are the circumstances when some or even all the vectors in an inner code set are identical.

Based on the above discussions, the RMS error of associative recollection can be generalized:

Definition 8 *Suppose M is an associative memory classifier built upon an inner code set $\{\vec{y}_k | k = 1, 2, \cdots, s\}$ which does not contain identical vectors. Also let $\{\tilde{\vec{x}}_k\}$ be the set of inspecting data with all the memorized patterns $\vec{x}_k$ $(k = 1, 2, \cdots, s)$ being represented, and*

$$\hat{\vec{y}}_k = F(\tilde{\vec{x}}_k M) \quad , \qquad k = 1, 2, \cdots, s \tag{3.13}$$

Then the general form of the associative recollection error of M inspected by $\{\tilde{\vec{x}}_k\}$ is:

$$\varepsilon(\{\vec{y}_k\}, \{\tilde{\vec{x}}_k\}) = f(\{\eta_k\}, \{\| \vec{y}_k - \hat{\vec{y}}_k \|\}) \tag{3.14}$$

where η_k is the appearance frequency of input pattern $\vec{x}_k$, $\| \vec{y}_k - \hat{\vec{y}}_k \|$ the absolute distance between $\vec{y}_k$, the inner code associated with $\vec{x}_k$, and $\hat{\vec{y}}_k$, the vector recollected by associative cue $\tilde{\vec{x}}_k$, and f an appropriate form of function.

Obviously, no matter what form f may take, it is the requirement of intuitiveness that there always exists:

$$\varepsilon \geq 0 \tag{3.15}$$

and:

$$f(\{\eta_k\}, \{\| \vec{y}_k - \vec{y}_k \|\}) = \min \varepsilon \tag{3.16}$$

These, of course, are satisfied by Eq.(3.12). Incidentally, it should be pointed out as well that in this thesis, for simplicity and generality, no specific assumptions will be made about the application of classifications afterwards. As a result, the input patterns will be chosen to be the data for inspection, and also the uniform appearance frequency assumed.

Now that the measurement of the associative recollection error of an AMC system is well-defined, it is straightforward to set up the criterion for the determination of an optimal inner code set:

Criterion 4 *A set of inner codes* $\{\vec{y}_k\}$ *is said to be optimum (under the inspecting data set* $\{\hat{\vec{x}}_k\}$*), if:*

$$\varepsilon(\{\vec{y}_k\}, \{\hat{\vec{x}}_k\}) = \min \varepsilon \tag{3.17}$$

3.3 Optimal Inner Coding Scheme

3.3.1 Existence of Optimal Inner Coding Schemes

In spite of the establishment of the optimum criterion for inner coding schemes, the existence of such a scheme or schemes need to be assured first before their seeking is started. This question can be answered by the following theorem.

Theorem 2 *In an associative memory classifier, if the output patterns are coded in the form of binary (or bipolar) vectors of limited dimensionality, then there always exists an optimal inner coding scheme in the sense of Criterion 4.*

Proof:

Presume M is the AMC designed to recognize a set of s patterns $\{\vec{x}_k\}$, each element of which is associated with a binary (or bipolar) vector $\vec{y}_k$ of fixed dimensionality n. Therefore, according to Eq.(3.6), this set of s row vectors can be represented by an $s \times n$ binary (or bipolar) code matrix Y. A specific coding scheme is reached each time the Y is evaluated.

Evidently, the total possibility of different Y is up to $2^{s \times n}$. Therefore, the possible inner coding schemes here is limited.

Now for each set of these inner codes, its associative recollection error can be estimated under one and the same group of inspecting data. And also, the total number of these εs obtained therefrom is limited. By simply applying a sorting algorithm, the minimum among these limited εs is attainable. Therefore, according to Criterion 4, its corresponding matrix Y or Ys is just the set of optimal inner codes.

Finally, it is stressed that as long as n is large enough so that:

$$s < 2^n \tag{3.18}$$

a set of pairwise different output vectors $\{\vec{y}_k\}$ $(k = 1, 2, \cdots, s)$ will be available.

That completes the proof of this theorem.

3.3.2 Complexity of Seeking the Optimal Inner Coding Scheme

As a matter of fact, the existence of optimal inner coding schemes is proved by the enumeration of all of their possibilities. No matter how many such schemes may be, its number is limited. For this reason, enumeration is effective in achieving such a theoretical conclusion. However, under any practical circumstances when a particular scheme needs to be worked out, enumeration is not a feasible way any more, for the complexity of this process will be $O(2^{s\times n}) = 2^{s\times n}$, which is computationally impractical.

It might be argued that the $2^{s\times n}$ schemes actually contain an enormous amount of trivial cases. To exclude these, the enumeration can be conducted in the following manner. The possible choice for the first vector out of the total s inner codes is 2^n, that for the second one is $2^n - 1$, *i.e.* every vector except what has been assigned to be the previous one is to be selected, and so on and so forth, until the last one which may have $2^n - s + 1$ possibilities. Consequently, the total number of schemes in this case is:

$$\begin{aligned} & 2^n(2^n-1)\cdots(2^n-s+1) \\ = \; & 2^{s\times n} - [1+2+\cdots+(s-1)]2^{(s-1)\times n} + \cdots + (-1)^{s-1}(s-1)!2^n \end{aligned} \quad (3.19)$$

Clearly, the reduction is not in the order of magnitude perspective. Therefore, this brings about no improvement in terms of the procedure's complexity.

Enumeration in reality does not provide any enlightenment on how to find a better or even optimal coding scheme. Our experience in the previous chapter shows

that usually a particular set of vectors is selected for some specific characteristics it possesses in its entirety. But strictly speaking, inner coding means assigning an inner code to each of the input patterns. In this sense, merely finding a set of vectors suitable to be the inner codes can at most accomplish the work partly. It still needs to indicate how to associate an input pattern with a particular vector in the inner code set, the process of which will be referred to as pairing. Whatever happens, this actually provides a practical way in finding inner codes which tend to be appropriate.

Based on the above strategy, inner coding is fulfilled in two steps by first choosing a set of proper vectors and then pairing. The first stage will be accomplished once and for all. Therefore, the complexity of the whole process depends on that of the pairing procedure. Here, because the forms of all these inner codes have already been fixed, enumerating all its possibility becomes a permutation problem, the result of which is $s!$, the factorial of s. However, since:

$$O(s!) = s^s \tag{3.20}$$

so according to Eq.(3.18), the complexity of the coding process under this strategy should still be estimated as:

$$s^s < (2^n)^s = 2^{s \times n} \tag{3.21}$$

Therefore, it can be concluded that:

Conclusion 3 *The complexity in seeking an optimal inner coding scheme through enumeration is $2^{s \times n}$, and therefore it is a non-polynomial (NP) problem.*

3.4 Better Inner Coding Scheme Seeking in Real Application

3.4.1 Inner Coding Scheme Seeking Strategy

During its information retrieval, the recollection product acquired from an AMC is a summation of terms all composed of vectors from three sources, that is, the associative cue, the input pattern, and its associated inner code (Eq.(3.3)). The last two forms jointly the weights of the network's neuron.

When the input patterns are to be applied to the network or, when the AMC is to be inspected by its input patterns, the recollecting relation of the whole system can be depicted by matrices in the manner of Eq.(3.9). Here, both the recollection product matrix and code matrix are just the list of row vectors of retrieval products and inner codes respectively. However, the system parameter matrix is constituted by the correlation number of each pair of input patterns or, that of an input vector and a vector from the inspecting data set.

The retrieval product matrix is the product of system parameter matrix and code matrix. The function of the nonlinear transformation is only to allow its operand being able to vary within a certain range instead of being a specific value. Yet, whether this can be achieved will depend on the retrieval result before its application. Since the input pattern is pre-defined upon the construction of an AMC, and so is the system parameter matrix, the only room left to us is the change of code matrix through the selection of inner codes. This just coincides with what has been obtained

in Conclusion 1 in the previous chapter from the observation of some special cases, and explains as well why AMC is also a data-driven system after the layout of its configuration.

In the light of the above discussion, either inner coding or, pairing if a set of vectors has already been chosen to be the inner codes, should be based on the characteristics of the system parameter matrix, so that the final recollected vectors acquired by applying nonlinear transform on the retrieval product will reduce or even minimize the associative recollection error. Here, pairing is actually a row vector reordering process of the code matrix.

In optimal inner coding scheme seeking, enumeration is intuitive but computationally infeasible. Generally speaking, the effective way to tackle an NP problem of this sort is to solve it on a case by case basis. Under present conditions, it is just to find the optimal coding or pairing scheme in accordance with the features of a given set of input patterns. Unfortunately, as has been pointed out in Section 2.5.1, the features of real-life data usually can not be described explicitly in mathematical expressions. Thus, what kind of strategy shall be taken?

So far, there still seems no systematic way in optimal inner coding scheme seeking for the data obtained in real-life. However, it has been demonstrated in the last chapter that a set of vectors with some desirable feature can be chosen as inner codes based on some statistical results of these data. Therefore, it is realistic to use the characteristics of the selected vectors to find out a better or optimal pairing strategy based on some assumptions about the system's parameters, and use this as a guidance

to achieve a better pairing scheme for the practical data.

In the previous chapter, Hadamard vectors are employed as inner codes in an AMC to recognize a group of multi-font Chinese characters. In the remaining part of this chapter, we shall use this as an example to demonstrate the efficiency of this pairing scheme improving strategy.

3.4.2 An Optimal Pairing Strategy When Hadamard Vectors Are Used as Inner Codes

In consideration of the properties of the original character vector, Hadamard vectors are chosen as associated output vectors in the AMC to recognize a group of multi font Chinese characters presented in Chapter 2. Hadamard vectors are the row vectors of a Hadamard transformation matrix. The latter has some specific qualities which are interesting to associative memory classifier.

First of all, it can be generated in a recursive way:

$$H_1 = [1] \tag{3.22}$$

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} H_1 & H_1 \\ H_1 & -H_1 \end{bmatrix} \tag{3.23}$$

and in general:

$$H_{2n} = \begin{bmatrix} H_n & H_n \\ H_n & -H_n \end{bmatrix} \tag{3.24}$$

Consequently, their dimensionalities, and therefore the numbers of Hadamard vectors available, are always the exponents of 2.

Secondly, the row vectors of these matrices are orthogonal to one another. Besides, all their corresponding components except the first one have exactly half the chance being both positive and negative.

Even after Hadamard vectors have been selected as the set of inner codes, the associative coding problem still remains unsolved yet, because it remains to be decided which input pattern should be paired with which. Previously the input and output vectors are paired simply according to the sequence of their natural appearance in each data set. Our next objective is to develop a pairing strategy which is optimal under some ideal conditions, and also may lead to better system performance of an AMC than that used formerly when applied to real-life data such as the set of original character vectors.

Suppose the system storage s is the exponent of 2, and the set of s-dimensional Hadamard vectors $\vec{h}_k$ $(k = 1,2,\cdots,s)$ are used as inner codes of the network. Let $t = \frac{s}{2}$, so the code matrix Y_s and the system parameter matrix C_s of the present network are as follows:

$$Y_s = \begin{bmatrix} \vec{h}_1 \\ \vec{h}_2 \\ \vdots \\ \vec{h}_s \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & \cdots & h_{1s} \\ h_{21} & h_{22} & \cdots & h_{2s} \\ \vdots & & & \\ h_{s1} & h_{s2} & \cdots & h_{ss} \end{bmatrix} = H_s \tag{3.25}$$

$$C_s = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1s} \\ c_{21} & c_{22} & \cdots & c_{2s} \\ \vdots & & & \\ c_{s1} & c_{s2} & \cdots & c_{ss} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} \tag{3.26}$$

where both C_s and Y_s are square matrices now, and C_{ij} $(i,j = 1,2)$ are all $t \times t$ block matrices:

$$C_{11} = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1t} \\ c_{21} & c_{22} & \cdots & c_{2t} \\ \vdots & & & \\ c_{t1} & c_{t2} & \cdots & c_{tt} \end{bmatrix}$$

$$C_{12} = \begin{bmatrix} c_{1\,t+1} & c_{1\,t+2} & \cdots & c_{1s} \\ c_{2\,t+1} & c_{2\,t+2} & \cdots & c_{2s} \\ \vdots & & & \\ c_{t\,t+1} & c_{t\,t+2} & \cdots & c_{ts} \end{bmatrix}$$

$$C_{21} = \begin{bmatrix} c_{t+1\,1} & c_{t+1\,2} & \cdots & c_{t+1\,t} \\ c_{t+2\,1} & c_{t+2\,2} & \cdots & c_{t+2\,t} \\ \vdots & & & \\ c_{s1} & c_{s2} & \cdots & c_{st} \end{bmatrix} \tag{3.27}$$

$$C_{22} = \begin{bmatrix} c_{t+1\ t+1} & c_{t+1\ t+2} & \cdots & c_{t+1\ s} \\ c_{t+2\ t+1} & c_{t+2\ t+2} & \cdots & c_{t+2\ s} \\ \vdots & & & \\ c_{s\ t+1} & c_{s\ t+2} & \cdots & c_{ss} \end{bmatrix}$$

and according to Eq.(3.24), Y_s in Eq.(3.25) can be repersented further below:

$$\begin{aligned} Y_s &= H_s = H_{2t} \\ &= \begin{bmatrix} H_t & H_t \\ H_t & -H_t \end{bmatrix} \end{aligned} \tag{3.28}$$

Now assume that the system coefficient matrix C_s also meets the following conditions:

(1) $c_{kl} > c_{kl'} > 0$, $k, l = 1, 2, \cdots, t; \quad l' = t+1, t+2, \cdots, s$

or $l' = 1, 2, \cdots, t; \quad k, l = t+1, t+2, \cdots, s$

(2) $c_{kk} > c_{kl}$, $k, l = 1, 2, \cdots, t$ where $l \neq k$

or $k, l = t+1, t+2, \cdots, s$ where $l \neq k$

(3) $c_{kl} = c_{k'l'}$, $k, l, k', l' = 1, 2, \cdots, t$ where $k \neq l,\ k' \neq l'$

or $k, l, k', l' = t+1, t+2, \cdots, s$ where $k \neq l,\ k' \neq l'$

or $k, k' = 1, 2, \cdots, t; \quad l, l' = t+1, t+2, \cdots, s$

or $l, l' = 1, 2, \cdots, t; \quad k, k' = t+1, t+2, \cdots, s$

Conditions (1) and (3) mean:

$$
\begin{aligned}
& c_{kl} = c_1 \ , \quad k,l = 1,2,\cdots,t && \text{where } k \neq l \\
\text{or} \quad & c_{kl} = c_3 \ , \quad k,l = t+1,t+2,\cdots,s && \text{where } k \neq l \ ; \\
\text{and} \quad & c_{kl} = c_2 \ , \quad \begin{cases} k = 1,2,\cdots,t \\ l = t+1,t+2,\cdots,s \end{cases} \\
\text{or} \quad & c_{kl} = c_4 \ , \quad \begin{cases} k = t+1,t+2,\cdots,s \\ l = 1,2,\cdots,t \end{cases} \ ; \\
\text{and} \quad & c_1 > c_2 \ , \quad c_3 > c_4
\end{aligned}
$$

where c_1, c_2, c_3, and c_4 are all positive constants. Besides, condition (2) is actually always satisfied since $c_{kk} = m$, the dimensionality of input vectors and $c_{kl} < m$, for all $k,l = 1,2,\cdots,s$ and $k \neq l$. Therefore:

$$
C_{11} = \begin{bmatrix} m & c_1 & \cdots & c_1 \\ c_1 & m & \cdots & c_1 \\ \vdots & & & \\ c_1 & c_1 & \cdots & m \end{bmatrix}_{t \times t} = (m - c_1)E_t + c_1 I_t \qquad (3.29)
$$

$$
C_{22} = \begin{bmatrix} m & c_3 & \cdots & c_3 \\ c_3 & m & \cdots & c_3 \\ \vdots & & & \\ c_3 & c_3 & \cdots & m \end{bmatrix}_{t \times t} = (m - c_3)E_t + c_3 I_t \qquad (3.30)
$$

$$C_{12} = \begin{bmatrix} c_2 & c_2 & \cdots & c_2 \\ c_2 & c_2 & \cdots & c_2 \\ \vdots & & & \\ c_2 & c_2 & \cdots & c_2 \end{bmatrix}_{t \times t} = c_2 I_t \tag{3.31}$$

$$C_{21} = \begin{bmatrix} c_4 & c_4 & \cdots & c_4 \\ c_4 & c_4 & \cdots & c_4 \\ \vdots & & & \\ c_4 & c_4 & \cdots & c_4 \end{bmatrix}_{t \times t} = c_4 I_t \tag{3.32}$$

where

$$E_t = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & & & \\ 0 & 0 & \cdots & 1 \end{bmatrix}_{t \times t} \tag{3.33}$$

$$I_t = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & & & \\ 1 & 1 & \cdots & 1 \end{bmatrix}_{t \times t} \tag{3.34}$$

are the identical matrix and the so-called all-one-element matrix of size $t \times t$ respectively.

Hence:

$$C_s = \begin{bmatrix} (m - c_1)E_t + c_1 I_t & c_2 I_t \\ c_4 I_t & (m - c_3)E_t + c_3 I_t \end{bmatrix} \tag{3.35}$$

and according to Eq(3.9):

$$\begin{aligned} Z_s &= C_s Y_s = C_s H_{2t} \\ &= \begin{bmatrix} (m-c_1)E_t + c_1 I_t & c_2 I_t \\ c_4 I_t & (m-c_3)E_t + c_3 I_t \end{bmatrix} \begin{bmatrix} H_t & H_t \\ H_t & -H_t \end{bmatrix} \\ &= \begin{bmatrix} [(m-c_1)E_t + (c_1+c_2)I_t]H_t & [(m-c_1)E_t + (c_1-c_2)I_t]H_t \\ [(m-c_3)E_t + (c_3+c_4)I_t]H_t & -[(m-c_3)E_t + (c_3-c_4)I_t]H_t \end{bmatrix} \\ &= \begin{bmatrix} Z_{11}H_t & Z_{12}H_t \\ Z_{21}H_t & Z_{22}H_t \end{bmatrix} \end{aligned} \tag{3.36}$$

Apparently, Z_{ij} $(i,j = 1,2)$ are all $t \times t$ matrices.

Now consider function F_M, the nonlinear transformation operated on matrices:

$$\begin{aligned} \hat{Y}_s &= F_M(Z_s) \\ &= F_M(C_s Y_s) \end{aligned} \tag{3.37}$$

or represented in the form of block matrices:

$$\begin{aligned} \hat{Y}_s &= \begin{bmatrix} \hat{Y}_{11} & \hat{Y}_{12} \\ \hat{Y}_{21} & \hat{Y}_{22} \end{bmatrix} \\ &= F_M\left(\begin{bmatrix} Z_{11}H_t & Z_{12}H_t \\ Z_{21}H_t & Z_{22}H_t \end{bmatrix}\right) \end{aligned} \tag{3.38}$$

and:

$$\hat{Y}_{ij} = F_M(Z_{ij}H_t) \quad , \qquad i,j = 1,2 \tag{3.39}$$

Although the original nonlinear transformation defined in Eq.(3.10) is in the form of vector, the operation itself, as can be seen from Eq.(3.11), is intrinsically applied

on the component basis. That is, it is actually defined on a numeral. The nonlinear function F_M in Eq.(3.37) expresses this one and the same transformation but operates on a matrix of any size. Therefore, when F_M degenerates to the case of operating on a single numeral, *i.e.* when M is a matrix of single element, it should be consistent with the operation defined in Eqs.(3.10) and (3.11). The unit value there is 1, and z_{kl}s can be interpreted as its coefficients. Consequently, if the operands of this nonlinear transformation are matrices, then the unit should be a matrix of certain size with the absolute value of all its elements to be 1, and the coefficient matrices of the same size. For instance, in Eqs.(3.38) and (3.39), the unit matrix is H_t, and coefficients Z_{ij}s are matrices of the same size.

Before a formal definition is framed for F_M, the following concepts need to be presented first:

Definition 9

1. *A matrix Z is positive, if and only if all of its elements are positive;*
2. *A matrix Z is negative, if and only if all of its elements are negative;*
3. *A matrix Z is non-positive, if and only if all of its elements are non-positive;*
4. *A matrix Z is non-negative, if and only if all of its elements are non-negative.*

Hence, we have:

Definition 10 *The nonlinear transformation function operated on matrices, F_M, for:*

$$\hat{Y} = F_M(ZH) \tag{3.40}$$

functions as follows:

$$\hat{Y} = \begin{cases} H & , \ \textit{if } Z \textit{ is positive} \\ -H & , \ \textit{if } Z \textit{ is negative} \\ \textit{non-defined} & , \ \textit{otherwise} \end{cases} \tag{3.41}$$

where H and Z are unit and coefficient matrices respectively.

Now, from Eq.(3.36):

$$\begin{aligned} Z_{11} &= (m - c_1)E_t + (c_1 + c_2)I_t \\ Z_{12} &= (m - c_1)E_t + (c_1 - c_2)I_t \\ Z_{21} &= (m - c_3)E_t + (c_3 + c_4)I_t \\ Z_{22} &= -[(m - c_3)E_t + (c_3 - c_4)I_t] \end{aligned} \tag{3.42}$$

Since:

$$c_1,\ c_2,\ c_3,\ c_4 > 0, c_1 > c_2, c_3 > c_4, \text{ and } I_t \text{ is positive}$$

so:

$$(c_1 + c_2)I_t, (c_1 - c_2)I_t, \text{ and } (c_3 + c_4)I_t, (c_3 - c_4)I_t \text{ are all positive}$$

also:

$$m > c_1, m > c_3, \text{ and } E_t \text{ is non-negative}$$

hence:

$$(m - c_1)E_t \text{ and } (m - c_3)E_t \text{ are non-negative}$$

therefore:

$$Z_{11}.\ Z_{21}.\ \text{and } Z_{12} \text{ are positive, while } Z_{22} \text{ is negative}$$

then, according to nonlinear function F_M (Eqs.(3.40) and (3.41)), the following is attained from Eq.(3.38):

$$\begin{aligned}
\hat{Y}_s &= F_M(C_s Y_s) = F_M(Z_s) \\
&= F_M\left(\begin{bmatrix} Z_{11}H_t & Z_{12}H_t \\ Z_{21}H_t & Z_{22}H_t \end{bmatrix}\right) \\
&= \begin{bmatrix} H_t & H_t \\ H_t & -H_t \end{bmatrix} \\
&= H_{2t} \\
&= Y_s
\end{aligned} \tag{3.43}$$

That is to say:

Theorem 3 *If the input patterns of an associative memory classifier can be arranged in an order so that its system parameter matrix satisfies the above-mentioned conditions (1) to (3), then the exact form of Hadamard vectors will be recollected under the application of input patterns to the network.*

The key to the proof of the above theorem is to determine the nature of the coefficient matrices Z_{11}, Z_{12}, Z_{21}, and Z_{22} in Eq.(3.36), *i.e.* whether they are positive or not, *etc.* With the help of condition (3), it is possible to express these coefficient matrices in terms of E_t and I_t (Eq.(3.42)), and thus easy to check for this feature. However, the conclusion of this theorem will hold, as long as Z_{11}, Z_{12}, Z_{21} are kept to be positive and Z_{22} negative. Now, from Eqs.(3.26) and (3.28), the recollection

product matrix can be expressed as:

$$\begin{aligned} Z_s &= C_s Y_s = C_s H_{2t} \\ &= \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} \begin{bmatrix} H_t & H_t \\ H_t & -H_t \end{bmatrix} \\ &= \begin{bmatrix} (C_{11}+C_{12})H_t & (C_{11}-C_{12})H_t \\ (C_{21}+C_{22})H_t & (C_{21}-C_{22})H_t \end{bmatrix} \\ &= \begin{bmatrix} Z_{11}H_t & Z_{12}H_t \\ Z_{21}H_t & Z_{22}H_t \end{bmatrix} \end{aligned} \tag{3.44}$$

That is, generally, the coefficient matrices have the following form:

$$\begin{aligned} Z_{11} &= C_{11} + C_{12} \\ Z_{12} &= C_{11} - C_{12} \\ Z_{21} &= C_{21} + C_{22} \\ Z_{22} &= C_{21} - C_{22} \end{aligned} \tag{3.45}$$

Here, each element in these coefficient matrices are the summation or subtraction of corresponding elements in either C_{11} and C_{12} or C_{21} and C_{22}. Clearly, it can be concluded that Z_{11}, Z_{12}, Z_{21}, and $-Z_{22}$ are all positive whenever condition (1) stands. Besides, condition (2) is automatically satisfied due to the nature of AMC. Therefore, the following is reached:

Corollary 4 *As long as the aforementioned condition (1) is satisfied in the system parameter matrix, the conclusion of Theorem 3 holds.*

Finally, under the above circumstances:

$$\hat{Y}_s = Y_s = H_s$$

(Eq.(3.43)), hence,

$$\hat{\vec{y}}_k = \vec{y}_k = \vec{h}_k$$

That is, in these conditions, the associative recollection error ε achieves its minimum according to Eqs.(3.15) and (3.16). Therefore, based on Criterion 4, this inner coding scheme is an optimal one.

We now summarise the above discussion as follows:

Theorem 5 *Suppose* $\{\vec{x}_k | k = 1, 2, \cdots, s\}$ *is a set of input patterns, where* s *is an exponent of* 2. *Assume also that* $\{\vec{h}_k\}$ *is a set of* s *Hadamard vectors generated recursively. If the input pattern set can be sorted in an order:*

$$\vec{x}_{k_1}, \vec{x}_{k_2}, \cdots, \vec{x}_{k_s}$$

so that the system parameter matrix:

$$C_s = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1s} \\ c_{21} & c_{22} & \cdots & c_{2s} \\ \vdots & & & \\ c_{s1} & c_{s2} & \cdots & c_{ss} \end{bmatrix}$$

$$= \begin{bmatrix} \vec{x}_{k_1}\vec{x}_{k_1}^T & \vec{x}_{k_1}\vec{x}_{k_2}^T & \cdots & \vec{x}_{k_1}\vec{x}_{k_s}^T \\ \vec{x}_{k_2}\vec{x}_{k_1}^T & \vec{x}_{k_2}\vec{x}_{k_2}^T & \cdots & \vec{x}_{k_2}\vec{x}_{k_s}^T \\ \vdots & & & \\ \vec{x}_{k_s}\vec{x}_{k_1}^T & \vec{x}_{k_s}\vec{x}_{k_2}^T & \cdots & \vec{x}_{k_s}\vec{x}_{k_s}^T \end{bmatrix}$$

obtained thereof satisfies:

$$c_{ij} > c_{ij'} > 0 \quad , \qquad i,j = 1,2,\cdots,t, \quad j' = t+1, t+2, \cdots, s$$
$$or \quad j' = 1,2,\cdots,t; \qquad i,j = t+1, t+2, \cdots, s \tag{3.16}$$

where $t = \frac{s}{2}$, *then* $\{\vec{h}_k | k = 1,2,\cdots,s\}$ *is a set of optimal inner codes under the pairing scheme of:*

$$\vec{x}_{k_i} \longleftrightarrow \vec{h}_i$$

that is, to associate each $\vec{x}_{k_i}$ *with the corresponding* $\vec{h}_i$.

3.4.3 Its Application in Real-life Data

Theorem 5 gives an optimal inner coding strategy when the system parameter matrix of a set of input patterns meets the condition described in Eq.(3.46). This actually provides an optimal pairing scheme when Hadamard vectors are utilized as inner codes. Although this condition is simple when expressed in inequalities, it may not be moderate in the view of a practical situation. For instance, it is unrealistic to believe that the set of 255 multi-font Chinese character images used formerly will be made to satisfy this condition since the data is composed of 85 different characters, each in three different fonts. On the other hand, what Eq.(3.46) requires is that the whole set of data will be grouped into two subsets, each containing about the same number of characters, and the input patterns formed thereof are very similar within each subset but at the same time quite different between them so that the correlation number of any two patterns in C_{11} and C_{22} will be greater than that in C_{12} and C_{21} of Eq.(3.26). Then how can this pairing strategy be applied to practical problems?

As a matter of fact, this pairing strategy stems from one of the major features of Hadamard transformation matrices, that is, such a matrix of order s can be derived from the Hadamard transformation matrix of order $t = \frac{s}{2}$ in the manner of Eq.(3.24). This makes it possible to express the retrieval product matrix Z_s in block matrices of Eq.(3.44) where the coefficient matrices are described in Eq.(3.45). However, this results in the dichotomy of the original data. Although this seems not to conform to our real situation, it is reasonable to believe that more likely this group of data can be divided into many small subgroups, so that the character images among each of which look more similar to one another than any characters of other groups. One of the straightforward ways to carry out the division is to dichotomize the original data group successively. This is just contrary to the recursive generation process of Hadamard transformation matrix in terms of data size. Therefore when depicted by system parameter matrix, the data grouped thereof comply easily with the code matrix, or the Hadamard transformation matrix in the present case, in the order of block matrices.

Now, let us have a further look at how this successive dichotomy may relax the condition in Theorem 5. For the time being, suppose the system parameter matrix in Eq.(3.26) has been divided into 16 block matrices of the same order:

$$C_s = \begin{bmatrix} D_{11} & D_{12} & D_{13} & D_{14} \\ D_{21} & D_{22} & D_{23} & D_{24} \\ D_{31} & D_{32} & D_{33} & D_{34} \\ D_{41} & D_{42} & D_{43} & D_{44} \end{bmatrix} \tag{3.47}$$

where each D_{ij} $(i,j = 1,2,3,4)$ is a $\frac{t}{2} \times \frac{t}{2}$ matrix. As a result, from Eq.(3.26), there are:

$$\begin{aligned} C_{11} &= \begin{bmatrix} D_{11} & D_{12} \\ D_{21} & D_{22} \end{bmatrix} \\ C_{12} &= \begin{bmatrix} D_{13} & D_{14} \\ D_{23} & D_{24} \end{bmatrix} \\ C_{21} &= \begin{bmatrix} D_{31} & D_{32} \\ D_{41} & D_{42} \end{bmatrix} \\ C_{22} &= \begin{bmatrix} D_{33} & D_{34} \\ D_{43} & D_{44} \end{bmatrix} \end{aligned} \tag{3.48}$$

Hence according to Eq.(3.45), Z_{12}, as an example of the coefficient matrices in Eq.(3.44), will be:

$$\begin{aligned} Z_{12} &= C_{11} - C_{12} \\ &= \begin{bmatrix} D_{11} & D_{12} \\ D_{21} & D_{22} \end{bmatrix} - \begin{bmatrix} D_{13} & D_{14} \\ D_{23} & D_{24} \end{bmatrix} \\ &= \begin{bmatrix} D_{11} - D_{13} & D_{12} - D_{14} \\ D_{21} - D_{23} & D_{22} - D_{24} \end{bmatrix} \end{aligned} \tag{3.49}$$

Obviously, for Z_{12} to be positive, it is enough if all of its four block matrices are positive. This requires all the elements in each block matrix of C_{11} to be greater than those in the corresponding block matrix of C_{12}. Therefore, instead of the requirement of Eq.(3.46) which states that each element in C_{11} should be greater than all those in C_{12}, there is no restriction any more on the elements of two block matrices positioned

differently in C_{11} and C_{12} respectively.

What does this mean to the practical data? In our current case, presume the 255 Chinese character images have been divided into two groups. Now, the dichotomies are conducted once more to get four subgroups: subgroups one and two from the first group, and subgroups three and four from the second group. Suppose there is a character image originally in the second group and now being divided into subgroup three which is more similar to some character images in the first group than some others in its own group. Of course, there are many other characters in the first group which look less like it. If the character images of this first group are dichotomized into two subgroups so that the similarities between any two character images of the same subgroup are always larger than those of any one of them with another character image from other subgroups, then the first block matrix in Eq.(3.49), $D_{11} - D_{13}$, is positive. That is to say, it does not matter even if some elements of D_{13} may be larger than those of D_{12}, as long as those of D_{11} is the largest. Also, assume that the character images in subgroup four are relatively less similar to any of those in the first group, then $D_{12} - D_{14}$ may be positive. This same kind of analysis applies to the remaining part of Z_{12}, and further to the other coefficient matrices. Since there are four data groups adopted presently other than the previous two, it is more likely that this grouped similarity nature may hold. Finally, these data groups can be dichotomized further according to the need.

In one word, in order to apply Theorem 5 to the real data, such as the set of 255 images of multi-font Chinese characters used formerly, for a better pairing scheme, the

original data set needs to be dichotomized successively. For this reason, the following data even-dividing algorithm has been devised:

Algorithm 1 *Data Set Dichotomy Algorithm:*

1. *For the set of data containing* s *pattern vectors,* $\Phi = \{\vec{x}_1, \vec{x}_2, \cdots, \vec{x}_s\}$, *calculate the correlation number* c_{kl} *of each pair of two different vectors* $\vec{x}_k$ *and* $\vec{x}_l$*:*

$$c_{kl} = \vec{x}_k \vec{x}_l^T \ , \quad k, l = 1, 2, \cdots, s \ \wedge \ k \neq l$$

2. *Select a pair of pattern vectors* $\vec{x}_{k_1}$ *and* $\vec{x}_{l_1}$, *which share the largest correlation number, and this forms the initial data in the first subgroup* ϕ_1, *namely:*

$$\phi_1 = \{\vec{x}_{k_1}, \vec{x}_{l_1}\}$$

where:

$$c_{k_1 l_1} \geq c_{kl} \ , \quad k \neq k_1 \ \vee \ l \neq l_1 \ , \quad 1 \leq k_1, l_1 \leq s \ , \ \textit{and} \ \ k, l = 1, 2, \cdots, s$$

3. *Selet the pattern vector* $\vec{x}_{k_2}$, *whose sum of correlation numbers with the two patterns chosen above is the smallest, and this is the first data in the second subgroup* ϕ_2, *namely:*

$$\phi_2 = \{\vec{x}_{k_2}\}$$

where:

$$c_{k_1 k_2} + c_{l_1 k_2} \leq c_{k_1 l} + c_{l_1 l} \ , \quad k_2, l = 1, 2, \cdots, s \ , \ \textit{and} \ \ k_2, l \neq k_1 \ , \ k_2, l \neq l_1$$

4. *For all the remaining pattern vectors in the original data set* Φ, *repeat the following while* $|\phi_1| < \frac{s}{2}$ *and* $|\phi_2| < \frac{s}{2}$, *i.e. neither of the two subgroups contains half of the total pattern vectors yet:*

(a) Take a new pattern vector $\vec{x}_k$, and calculate its average correlation numbers c_1 and c_2 with all the pattern vectors already in subgroups ϕ_1 and ϕ_2 respectively:

$$\bar{c}_1 = \frac{1}{|\phi_1|} \sum_{\forall \vec{x}_l \in \phi_1} c_{kl}$$

$$\bar{c}_2 = \frac{1}{|\phi_2|} \sum_{\forall \vec{x}_l \in \phi_2} c_{kl}$$

where $|\phi_1|$ and $|\phi_2|$ represent the numbers of elements in sets ϕ_1 and ϕ_2 separately,

(b) Assign pattern vector $\vec{x}_k$ to the subgroup with the patterns of which it has a larger average correlation number, that is:

$$\begin{cases} \phi_1 \longleftarrow \phi_1 \cup \{\vec{x}_k\} \\ \phi_2 \longleftarrow \phi_2 \end{cases} \quad \text{if} \quad \bar{c}_1 > \bar{c}_2$$

$$\text{or} \quad \begin{cases} \phi_1 \longleftarrow \phi_1 \\ \phi_2 \longleftarrow \phi_2 \cup \{\vec{x}_k\} \end{cases} \quad \text{if} \quad \bar{c}_1 < \bar{c}_2$$

5. Put all the pattern vectors left into another subgroup:

$$\phi_2 = \Phi - \phi_1, \quad \text{if} \quad |\phi_1| = \frac{s}{2}$$

$$\text{or} \quad \phi_1 = \Phi - \phi_2, \quad \text{if} \quad |\phi_2| = \frac{s}{2}$$

3.5 Simulation on the Performance of different Pairing Schemes

3.5.1 Testing Data and the Recognition Criteria

The applicability of associative memory network as pattern classifier has been manifested preliminarily in the previous chapter with a set of 85 real collected Chinese characters in three different fonts, namely, Song, Kai, and boldface. For the sake of convenience in comparison, this same group of data is utilized here once again to test the efficacy of different pairing schemes when Hadamard vectors have been chosen as inner codes in the associative memory classifier. Therefore, totally 255 character patterns are involved in the experiments below, and the same number of Hadamard vectors with their dimensionalities being 256 are the class vectors to be associated with them.

The same six groups of testing data, each generated from all the 255 character images as before, are used again. The first group is the original charater vectors themselves, while in groups two to four, noise patterns are produced with ten, twenty, and forty percent of randomly selected vector components reversing their initial states, that is, their values are changed from 1 to -1, and *vice versa*. Finally, in groups five and six, a horizontal white, or 'background', and black, or 'character' stripes are laid over the central part of every character picture separately. These same testing data groups are utilized here once more merely for the simplicity in comparison. Some character images of these testing data have been shown in Figure 2.7.

The set of recognition criteria stated in Section 2.6.1, which contains a classifi

cation, a rejection, and a correctness rule, is employed here again. The AMCs to be established will be different from one another, and from the previous one only in the change of pairing partner of input and output pattern vectors, while the input patterns will still be the formerly used set of original character vectors. Therefore, the misclassification ruling to the recollection result of different fonts of the same character and the rejection ruling to the outcome with different fonts of the same character sharing the largest similarity value, although both are actually acceptable in the perspective of multi-font character recognition, still hold.

3.5.2 Results and Analyses

Hadamard vectors have been chosen as inner codes in the last chapter, but in the AMC there, pairing is conducted based on the natural order of every original character vector and Hadamard vector in their own data sets respectively. The recognition behaviour of this AMC has been displayed in both Tables 2.2 and 2.3.

Now, it is attempted to find some better pairing schemes by dichotomizing the primitive set of original character vectors successively into many smaller subgroups. Correspondingly, the Hadamard vector set will also be divided sequentially into many subgroups of the same size. The input and output pattern subgroups are then associated in sequence. However, within each of these subgroup pairs, the pairing of every individual input and output pattern is still in agreement with their appearance. Therefore, for the entire data set, pairing among subgroups is determined, but that of the data within every subgroup still remains to be in random unless each subgroup

contains solely one element.

The classification results of the first testing data group, that is, the set of original character vectors, by the AMCs built up when pairing is carried out only after the input pattern set is divided first into 2, 4, 8, ..., through 64 subgroups separately are illustrated in Table 3.1. The numbers of misclassification (above) and rejection (below) are listed first in accordance with each font, then added together for the entire data set. The recognition, rejection, and error rates of the latter are followed thereafter. For the sake of comparison purpose, the results in the case of non-dichotomy of input data set, that is, the case of our simulation in Chapter 2, are also listed. This occurs when the total number of subgroups is 1.

In addition, Table 3.2 gives the maximal, the minimal, and the average values of LSV of each type of font and the entire data set in accordance with every pairing scheme, since the classification is achieved on the basis of LSV. It has been pointed out in Section 2.6.3 that LSV can be used as a confidence indication of an AMC's classification process. This becomes even clearer after the introduction of associative recollection error ε of an AMC system (Section 3.2). ε is measured based on the distances between each output pattern and the vector recollected by the input pattern it is associated with. Since the calculation of distance and similarity between two vectors are just complementary operations, these LSVs, particularly the average one, are the very useful approximants obtainable during a real classification process to the theoretical estimation. As a matter of fact, especially in the development of AMC, such statistical measurements as the average LSV of a set of data are under

	Grouping Status (The No. of Subgroups)						
	1	2	4	8	16	32	64
Song	0	2	3	1	0	0	0
	0	3	0	0	1	1	0
Kai	1	4	1	3	1	1	1
	0	2	1	0	1	0	2
Boldface	1	0	0	1	1	0	0
	0	2	0	0	0	0	0
Entire Set	2	6	4	5	2	1	1
	0	7	1	0	2	1	2
Recognition Rate	99.22%	94.90%	98.04%	98.04%	98.43%	99.22%	98.82%
Rejection Rate	0.0%	2.75%	0.39%	0.0%	0.78%	0.39%	0.78%
Error Rate	0.78%	2.35%	1.57%	1.96%	0.78%	0.39%	0.39%

Table 3.1: The classification results of the original character patterns from the AMCs where pairing is conducted after first dividing them into 1, 2, 4, ..., through 64 subgroups separately.

		Grouping Status (The No. of Subgroups)						
		1	2	4	8	16	32	64
Song	maximum	196	201	208	206	208	204	204
	minimum	153	151	148	153	154	155	147
	average	175.07	176.36	181.04	181.06	180.68	182.07	181.40
Kai	maximum	193	187	194	188	190	187	188
	minimum	149	148	147	149	152	151	149
	average	169.46	167.42	172.18	173.09	171.61	172.52	171.06
Boldface	maximum	204	210	206	211	207	211	215
	minimum	163	156	161	168	163	164	166
	average	187.74	189.67	190.33	191.59	191.66	191.78	191.56
Entire Set	maximum	204	210	208	211	208	211	215
	minimum	149	148	147	149	152	151	147
	average	177.42	177.82	181.18	181.91	181.32	182.12	181.34

Table 3.2: The statistical data of LSV obtained during the information retrieval process from the AMCs where pairing is conducted after first dividing the original character patterns into 1, 2, 4, ..., through 64 subgroups separately.

no circumstances less important than the deterministic ones like recognition rate *etc.*

The results in both Tables 3.1 and 3.2 reflect the general tendency that the AMCs' performances become better when the input patterns are grouped smaller before pairing. Besides in Table 3.2, almost all the average LSVs in the pairing schemes with dichotomous processing of the primitive data set are larger than their counterparts in column 1, with the only exception which occurs to font Kai when the whole set of data is merely halved. That is, the strategy discussed in the previous section does offer better pairing schemes than that used in the last chapter. Nevertheless, this seems not true with regard to the recognition rates shown in Table 3.1. On the contrary, a decline is observed particularly when the sizes of subgroups are not small enough, with the outcome in column 2 as a notable example. Why has this been the case?

First of all, the AMCs' performances may degenerate slightly at least measured determinatively due to the impairment of anti-noise effect of data randomness when the data set dichotomy algorithm is applied abruptly. Just as the appearance of a certain input pattern in the data set is totally random, so is that of every correlation number $c_{kl} = \vec{x}_k \vec{x}_l^T$ in the system parameter matrix C. Therefore, according to Eq.(2.3), the crosstalk caused by all the other input patterns tend to counterbalance against one another during information retrieval when the system coding scheme is totally based on the patterns appearance in the data set. However, this effect may be weakened severely when the data set dichotomy algorithm is applied suddenly, since the purpose of this algorithm is nothing but to make the coefficient numbers appear in a certain order.

Secondly, when not only the correct recognition but also the rejection and misclassification are observed jointly, it is found that part of the improvement in AMCs' behaviour is concealed by the multi-font nature of the data in use. For instance, in the pairing schemes of 16, 32, and 64 divided subgroups, part of the rejection and more importantly all the misclassification are caused by different fonts of the same character. Nonetheless, in the previous experiment when pairing was conducted without data set division, there was still misclassification among different characters.

In short, except some degeneration when the input pattern set is divided evenly for the first few times, generally speaking, the AMCs' performance gets better as this process goes on. This can also be seen from the results of our further experiments described below.

The system performances of AMCs established by pairing after first grouping the 255 original character vectors into 16, 32, and 64 subsets separately are tested by all six groups of testing data. The classification results are presented in Tables 3.3 to 3.5, and the corresponding LSVs in Tables 3.6 to 3.8. All the information is shown in the same manners as before. Obviously, under such pairing schemes, the AMCs' classification behaviour attains an improvement since all the recognition rates of testing data groups two, three, four, and six in Tables 3.3 through 3.5 are increased, and there is no decrease in testing data group five, compared with their counterparts in Table 2.2. This can also be seen from Tables 3.6 through 3.8, where well over two thirds of the maximal and minimal LSVs and all the average LSVs are larger than their corresponding values in Table 2.3.

	Testing Data Group					
	I	II	III	IV	V	VI
Song	0	0	0	4	1	3
	1	0	0	1	1	3
Kai	1	1	1	5	3	3
	1	1	0	1	1	1
Boldface	1	1	0	1	2	1
	0	0	0	0	0	0
Entire Set	2	2	1	10	6	7
	2	1	0	2	2	4
Recognition Rate	98.43%	98.82%	99.61%	95.29%	96.86%	95.69%
Rejection Rate	0.78%	0.39%	0.0%	0.78%	0.78%	1.57%
Error Rate	0.78%	0.78%	0.39%	3.92%	2.35%	2.75%

Table 3.3: e classification results of all the six testing data groups from the AMC where the original character vectors are divided into 16 subgroups before pairing.

	Testing Data Group					
	I	II	III	IV	V	VI
Song	0	0	0	1	2	1
	1	0	0	3	1	3
Kai	1	1	1	4	2	2
	0	2	1	2	1	2
Boldface	0	0	0	0	1	0
	0	0	0	1	1	0
Entire Set	1	1	1	5	5	3
	1	2	1	6	3	5
Recognition Rate	99.22%	98.82%	99.22%	95.69%	96.86%	96.86%
Rejection Rate	0.39%	0.78%	0.39%	2.35%	1.18%	1.96%
Error Rate	0.39%	0.39%	0.39%	1.96%	1.96%	1.18%

Table 3.4: The classification results of all the six testing data groups from the AMC where the original character vectors are divided into 32 subgroups before pairing.

	Testing Data Group					
	I	II	III	IV	V	VI
Song	0	0	0	2	0	1
	0	0	0	1	0	0
Kai	1	3	2	4	2	7
	2	0	0	1	2	2
Boldface	0	0	0	1	1	0
	0	0	1	0	0	0
Entire Set	1	3	2	7	3	8
	2	0	1	2	2	2
Recognition Rate	98.82%	98.82%	98.82%	96.47%	98.04%	96.08%
Rejection Rate	0.78%	0.0%	0.39%	0.78%	0.78%	0.78%
Error Rate	0.39%	1.18%	0.78%	2.75%	1.18%	3.14%

Table 3.5: The classification results of all the six testing data groups from the AMC where the original character vectors are divided into 64 subgroups before pairing.

		Testing Data Group					
		I	II	III	IV	V	VI
Song	maximum	208	211	206	188	207	196
	minimum	154	151	155	146	150	151
	average	180.68	180.15	179.73	170.34	175.76	176.12
Kai	maximum	190	187	187	180	183	190
	minimum	152	153	156	148	152	152
	average	171.61	171.81	171.75	163.56	166.98	168.07
Boldface	maximum	207	209	206	191	204	202
	minimum	163	161	164	154	157	162
	average	191.66	191.39	189.60	177.06	185.13	186.95
Entire Set	maximum	208	211	206	191	207	202
	minimum	152	151	155	146	150	151
	average	181.32	181.12	180.36	170.32	175.96	177.05

Table 3.6: The statistical data of LSV obtained during the information retrieval processes of all the six testing data groups when the AMC is established by dividing the original character patterns into 16 subgroups before pairing.

		Testing Data Group					
		I	II	III	IV	V	VI
Song	maximum	204	205	203	190	200	194
	minimum	155	155	154	148	154	151
	average	182.07	180.65	179.86	170.24	175.47	175.78
Kai	maximum	187	186	190	179	180	183
	minimum	151	152	152	149	149	151
	average	172.52	172.06	172.18	163.58	166.22	167.49
Boldface	maximum	211	209	208	191	204	206
	minimum	164	161	164	155	150	163
	average	191.78	190.56	188.99	176.49	185.02	187.20
Entire Set	maximum	211	209	208	191	204	206
	minimum	151	152	152	148	149	151
	average	182.12	181.09	180.34	170.10	175.57	176.82

Table 3.7: The statistical data of *LSV* obtained during the information retrieval processes of all the six testing data groups when the AMC is established by dividing the original character patterns into 32 subgroups before pairing.

		Testing Data Group					
		I	II	III	IV	V	VI
Song	maximum	204	206	203	189	199	195
	minimum	147	150	157	151	147	149
	average	181.40	180.78	180.21	170.41	176.62	175.41
Kai	maximum	188	187	191	176	184	183
	minimum	149	155	152	149	147	152
	average	171.06	170.99	171.33	163.41	165.99	166.68
Boldface	maximum	215	215	208	194	202	209
	minimum	166	165	161	157	153	162
	average	191.56	191.82	189.58	175.93	184.86	188.25
Entire Set	maximum	215	215	208	194	202	209
	minimum	147	150	152	149	147	149
	average	181.34	181.20	180.37	169.92	175.82	176.78

Table 3.8: The statistical data of LSV obtained during the information retrieval processes of all the six testing data groups when the AMC is established by dividing the original character patterns into 64 subgroups before pairing.

Pairing Scheme	No. of Subgroups	16	32	64
	Size of Subgroups	16	8	4
Average Recognition Rate		97.45%	97.78%	97.84%
Average Rejection Rate		0.72%	1.18%	0.59%
Average Error Rate		1.83%	1.05%	1.57%
Average of Average *LSV*		177.69	177.67	177.57

Table 3.9: The averages of the recognition, the rejection, and the error rate, and of the average *LSV* of the entire data set over all six testing data groups in pairing schemes of dividing the input pattern set first into 16, 32, and 64 subgroups respectively.

Besides, a comparison of recognition rates among Tables 3.3 to 3.5 suggests a tendency of classification error increase if the original data set has been overdivided. Each input pattern subgroup in these three pairing schemes may have up to 16, 8, and 4 original character vectors respectively. The averages of their recognition, rejection, and error rate, and of the average *LSV* of the whole data set over all six testing data groups are shown in Table 3.9 in accordance with each of these pairing schemes. Clearly, after the input pattern set has been divided into 16 subgroups, there seems no improvement in terms of *LSV* if the data set dichotomy continues. However, if all these data subgroups are halved at this moment, there is a 0.33% increase in the average recognition rate. Also observed is an increase in the average rejection rate, and a drop in average error rate. Nevertheless, if the data subgroups are halved once more, there will be an increase in the average recognition rate of only

0.06%, but at a cost of a 0.52% increase in misclassification. In addition, under the testing data groups two and three, the cases in which recognition rates remain high regardless of heavy contaminations suffered, all the misclassifications are caused by the multi-font nature of the character data in pairing schemes of 16 and 32 divided subgroups. However, some of those will be made among different characters if pairing is conducted with further dichotomy on this set of data.

In a practical problem, it seems impossible for the system parameter matrix to satisfy the condition of Theorem 5, not even the relaxed one discussed in Section 3.4.3 completely. For this reason, the noise elimination function of data randomness is still necessary when the input patterns and inner codes are associated under these pairing schemes. The larger the size of data subgroup is, the better this anti-noise effect may be. On the other hand, it is also under the practical consideration that the system is expected to perform better as the input pattern set is dichotomized continuously. These two factors function jointly, and a proper size of data subset is to be reached. In our present case, it seems that the optimal size is 8, *i.e.* when the 255 original character vectors are divided evenly into 32 subgroups, under a comprehensive consideration of both the statistical and the deterministic measuring results.

3.6 Discussion

It has been known now that an associative memory network can be turned into a pattern classifier provided that an appropriate selection of the output vectors, or

inner codes, associated with all the patterns to be recognized is made. As a matter of fact, this inner coding problem has become the central issue in the construction of an associative memory classifier. Practically, it can be fulfilled in a two phase manner, *i.e.* choosing first a proper set of vectors with the desired features as the inner codes, and then working out a pairing scheme to associate each input pattern with a certain output vector. In Chapter 2, Hadamard vectors have been used specifically as a set of desired inner codes. And this chapter shows how a specially developed pairing strategy may lead to the improvement of AMCs' recognition performances.

So far, our discussion is based on the assumption that the form of input patterns has been pre-defined and is not subject to change. This, of course, is true when pattern classification is considered solely. However, in a real pattern recognition system, usually there is a feature extraction process right before that. Whatever the task it may have, normally it will result in a change in the data format of the input vectors. As has been discovered before, an AMC's performance relies in the final analysis on the characteristics of its input patterns. Therefore, is it possible for this to be taken into account in feature extraction so that the finally generated feature patterns are more suitable to such a kind of neural classifiers? This would be our major focus in the next chapter.

Chapter 4

Feature Extraction in Cooperation with Associative Memory Classifier

4.1 Introduction

Associative memory classifier is under development for its application in character recognition being kept in the background. The process of character recognition, like all the other pattern recognition problems, consists of two major stages, namely, feature extraction and pattern classification. Feature extraction is a critical issue for any recognition process since the final classification results depend to a large extent on it.

Traditionally, there are two fundamental objectives in the feature extraction process. The first is the determination of certain attributes of the pattern classes which are invariant to as many kinds of distortions as possible. The next is to reduce the dimensionality of the feature vector by selecting the most effective or discriminatory

characteristics of these patterns [117].

However, when neural network is used as a pattern classifier, a change occurs in these basic tasks of feature extraction. Although the selection of stable features still remains as one of its major objectives, data reduction is not a necessity any longer since the parallelism nature of neural network systems makes them fit for the processing of large amounts of data. For instance, in the back propagation model, the neural network most commonly used so far in character recognition, the time-consuming nature is one of the built-in defects in its learning process. This is caused in essence by the large number of presentations of the training data required for the convergence of this process [29]. Also, in certain cases of its application, such as handwritten zipcode recognition, the need for large scale training has made the situation even worse [35], [41]. However, all these have nothing to do with the dimensionality of the data. Different methods have been proposed to overcome this difficulty. They are modifications of either the configuration of the network [41], or certain aspects in the learning algorithm [35], or even the so-called learning environment in which the order, number, and repetition of training samples are concerned [59].

Although a higher dimensionality of training pattern may require more time in the simulation work conducted on a serial computer, this is merely a problem resulted from the constraint of the simulation environment rather than an essential characteristic of the neural network methods. Thus, a high dimensional feature vector can still be regarded as appropriate as long as it contains enough information to differentiate a pattern from all the others. Nevertheless, instead of its dimensionality, here the

suitability of the feature pattern vectors for the neural network classifier should be taken into account. This is because neural network becomes completely a data-driven system after the formation of its configuration, consequently the recognition performance of the neural classifier will be determined ultimately by the properties of these feature vectors under this circumstance.

In our study on associative memory classifier in the previous two chapters, excellent recognition performance on a set of multi-font Chinese characters has been achieved with strong resistance to random noise. However, since no specific feature extraction process was considered, it was not suitable for the recognition of some geometrically distorted characters. In this chapter, a feature extraction method applicable to associative memory classifier is proposed. The features obtained therefrom are free of translation and rotation, the two most common kinds of distortions which occur in printed character recognition. In the text below, the feature extraction algorithms are described first in Section 4.2. Discussed then in Sections 4.3 and 4.4 are some details in realizing the principle of geometrical deformation invariance and pursuit of the suitability of feature vectors to the neural network classifier. Finally, the effectiveness of this feature extraction process is demonstrated in both Sections 4.5 and 4.6 by a series of experiments in different perspectives.

4.2 Algorithms for Feature Extraction

4.2.1 Feature Extraction Algorithm FPFB

Assume a character image is represented by a two dimensional function $f(x, y)$:

$$f(x,y) = \begin{cases} 1, & \text{if the pixel at } (x,y) \text{ is on the character} \\ 0, & \text{if the pixel at } (x,y) \text{ is on the background} \end{cases} \tag{4.1}$$

where (x, y) are the discrete rectangular coordinates.

The distortions which occur most frequently in the recognition of printed characters are the translation and rotation of character images. Aiming at the extraction of feature vectors which are invariant to these distortions and accommodated to the AMC developed in our previous work, a feature extraction algorithm, called FPFB Algorithm, has been designed. As described below, it is a process of Fourier transform (a two dimensional one), followed by Polarization (the replacement of coordimate systems from a rectangular one to a polar one), another Fourier transform (a one dimensional one), and Binarization.

Algorithm 2 *FPFB Algorithm:*

1. *Conduct a two dimensional Fourier transform on the original binary character image* $f(x, y)$*:*

$$F(u,v) = \mathcal{F}_{(x,y)}(f(x,y)) \tag{4.2}$$

where $\mathcal{F}_{(x,y)}$ *denotes the two dimensional Fourier transform implemented on coordinates* x *and* y*:*

2. *Replace the Cartesian coordinate system of the character's Fourier spectrum* (u, v) *by a polar one* (ρ, θ)*:*

$$\begin{aligned} G(\rho, \theta) &= |F(u(\rho, \theta), v(\rho, \theta))| \\ &= |F(\rho \cos\theta, \rho \sin\theta)| \end{aligned} \tag{4.3}$$

3. *Conduct another one dimensional Fourier transform on this new spectrum* $G(\rho, \theta)$ *along the axis of polar angle* θ*:*

$$H(\rho, \phi) = \mathcal{F}_\theta(G(\rho, \theta)) \tag{4.4}$$

4. *Binarize each component of the feature patterns obtained thereof based on the values of the corresponding components of the whole group of patterns involved.*

One of the functions of this algorithm, the geometrical deformation invariant signal extraction, is fulfilled in its first three steps. This can be seen in the following discussion.

According to step one, $|F(u, v)|$, the Fourier spectrum of $f(x, y)$, is obtained first. Fourier transform has many useful properties, one of which is that the shift in the original image does not affect its magnitude. Therefore, $|F(u, v)|$ is translation invariant. Moreover, the rotation of the image is preserved in Fourier transform. That is, if

$$\mathcal{F}_{(\gamma,\alpha)}(f(\gamma, \alpha)) = F(\rho, \theta),$$

then

$$\mathcal{F}_{(\gamma,\alpha)}(f(\gamma, \alpha + \alpha_0)) = F(\rho, \theta + \alpha_0).$$

Here, both (γ, α) and (ρ, θ) are polar coordinates. This means rotational distortion, if there is any, will be kept exactly the same in $|F(u,v)|$ as in the original image $f(x,y)$ without any additional deformations introduced through the Fourier transform.

Step two is a change of coordinate systems, by which $|F(u,v)|$ becomes $G(\rho,\theta)$. In the image with discrete *Cartesian* coordinates $|F(u,v)|$, each pixel represents the image function value over a small area $\Delta u \Delta v$, while in $G(\rho,\theta)$ of discrete polar coordinates, the area element becomes $\rho \Delta \rho \Delta \theta$. Therefor , $|F(u,v)|$ and $G(\rho,\theta)$ are two functions although both are depicted in the form of a matrix.

Again, this change does not add any deformation to the original rotative distortion. That is, if the original character image is rotated with an angle α_0, and so is its Fourier spectrum, then the image function obtained in this step is $G(\rho, \theta + \alpha_0)$. This indicates that such a change in coordinate systems can transform the rotation into a one dimensional translation along the axis of polar angle. Therefore, by implementing another one dimensional Fourier transform on it in step three, $|H(\rho,\phi)|$, the new character spectrum obtained, is free from the translation of polar angle, that is, the rotation of the original character image. Consequently, $|H(\rho,\phi)|$ is both translation and rotation invariant.

4.2.2 Threshold Function Generation Algorithm

The original character pattern $f(x,y)$ is a binary image. Yet, its transformed spectra after each step, *i.e.* $|F(u,v)|$, $G(\rho,\theta)$, and $|H(\rho,\phi)|$, are all continuous functions of their respective discrete coordinates. In order to make the feature vector suitable

for the AMC previously developed, the distortion invariant signal $|H(\rho,\phi)|$ will be binarized to obtain a bipolar feature vector $h(\rho,\phi)$. Such an additional operation in feature extraction will offer another advantage, *i.e.* an increase in the ability of the whole process to resist computational errors accumulated in the above series of transformations.

The binarization is based on the transformed functions of the entire group of data. Suppose $\{f_k(x,y)|k = 1,2,\cdots,s\}$ is the group of characters under consideration. Then after the first three steps of operation, a group of distortion invariant feature patterns $\{|H_k(\rho,\phi)|\,|k = 1,2,\cdots,s\}$ is available. The problem now is to construct an appropriate threshold function $T(\rho,\phi)$, so that the binarization can be conducted on this ground. Obviously, different threshold functions may result in different patterns of feature vectors obtained from the binarization conducted thereof. A proper selection of this function will lead to the acquisition of a set of feature data which can drive the currently used neural network classifier to act more effectively, which happens to be the second objective of our feature extraction process.

According to the characteristics of AMC, the input patterns can be recognized more accurately if they are more likely to be orthogonal to one another. This can be achieved heuristically under bipolar coding by choosing the threshold function in such a way that, for each pair of specific coordinates (ρ,ϕ), T will divide about half the number of $|H_k(\rho,\phi)|$s into 1 and the rest -1. Another criterion in the construction of the threshold function is the robustness of those $|H_k(\rho,\phi)|$s near the threshold. The larger the smallest difference between the two $|H_k(\rho,\phi)|$s on the two sides of the

threshold, the better.

Based on these two criteria, an algorithm for the generation of threshold function has been devised as follows:

Algorithm 3 *Threshold Function Generation Algorithm:*

1. *For a certain (ρ, ϕ), divide the group of real-numbered features $|H_k(\rho, \phi)|$ $(k = 1, 2, \cdots, s)$ into two sets S_1 and S_2 so that, each element in S_1 is greater than any of those in S_2, and the difference in the number of elements in these two sets is as small as possible;*

2. *Calculate the means m_1 and m_2 within S_1 and S_2 separately;*

3. *Compute the quasi standard deviations σ_1 and σ_2 within S_1 and S_2 defined below:*

$$\sigma_1 = \sqrt{E(|H_i(\rho,\phi)| - m_1)^2} \qquad \forall\, |H_i(\rho,\phi)| \in S_1 \wedge |H_i(\rho,\phi)| < m_1,$$
$$\sigma_2 = \sqrt{E(|H_j(\rho,\phi)| - m_2)^2} \qquad \forall\, |H_j(\rho,\phi)| \in S_2 \wedge |H_j(\rho,\phi)| > m_2; \quad (4.5)$$

4. *Sequence those $|H_k(\rho, \phi)|$ satisfying the condition $\sigma_1 > |H_k(\rho, \phi)| > \sigma_2$, and the threshold $T(\rho, \phi)$ is the average of two consecutive $|H_k(\rho, \phi)|$s which differ the most;*

5. *Repeat steps 1 to 4 for every pair of coordinates.*

An example of the whole feature extraction process applied to a printed Chinese character is given by Figures 4.1 and 4.2, with the original image $f(x, y)$ shown in Figure 4.1(a). Its transformed spectra, $|F(u, v)|$, $G(\rho, \theta)$, and $|H(\rho, \phi)|$ are depicted

(a) (b)

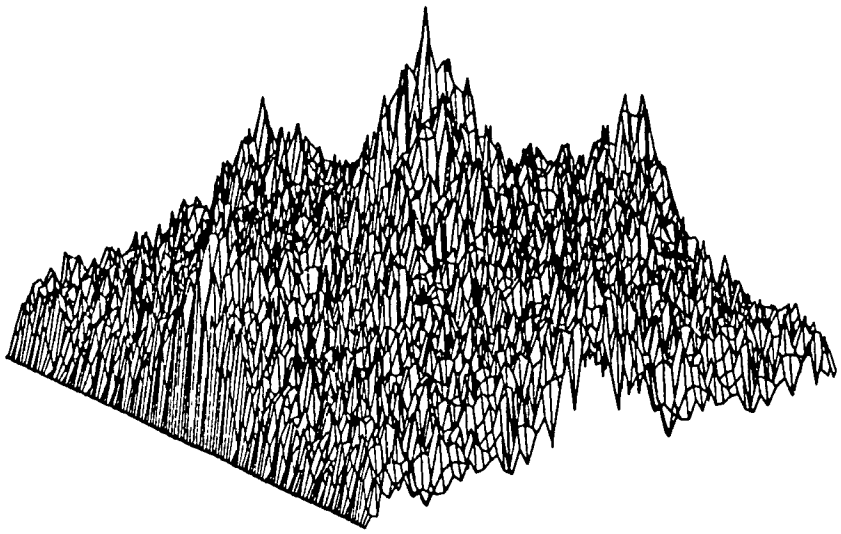
Figure 4.1: An example of extracting features from Chinese characters: the original character image (a), and its extracted bipolar feature pattern (b).

in (a) through (c) of Figure 4.2, and Figure 4.2(d) illustrates the threshold function $T(\rho, \phi)$ obtained from the set of 85 common Chinese characters in Appendix A, each of which is printed in three different fonts. The finally extracted binary feature pattern $h(\rho, \phi)$ is displayed in Figure 4.1(b).

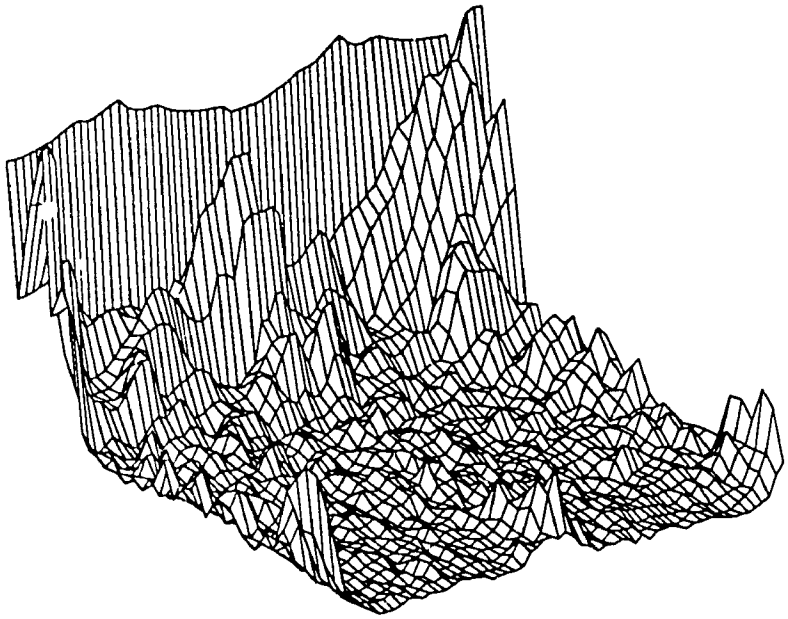
4.3 Replacement of Discretized Coordinate Systems

Since it has been proven theoretically that the Fourier spectrum is free of translation in the original image, the key point in the FPFB algorithm's geometrical deformation removal mechanism is to turn rotation into a one dimensional translation, so that it can be removed subsequently by a Fourier transform on that dimension. To achieve this, a change of coordinate systems is needed, as in step two of the FPFB algorithm. If the functions being dealt with are on continuous coordinates, this is simply the replacement of their arguments in the following manner:

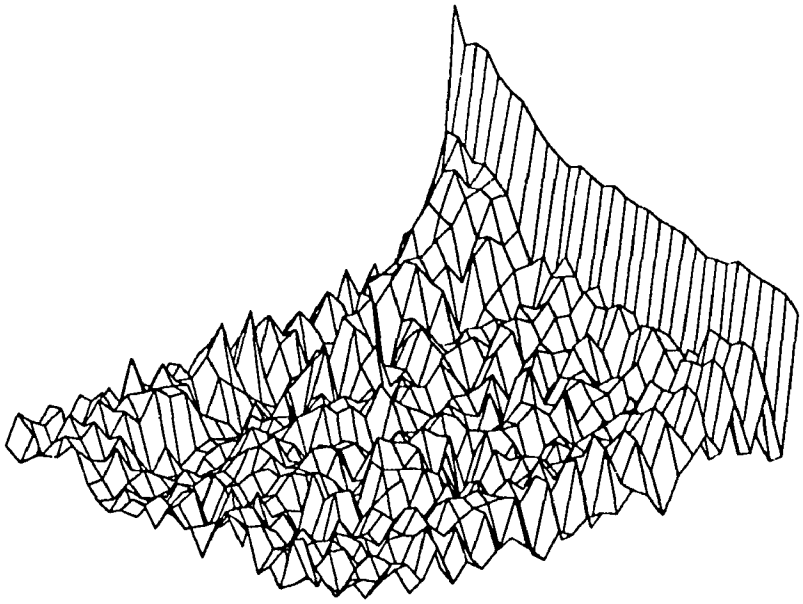
$$\begin{aligned} u &= \rho \cos\theta \\ v &= \rho \sin\theta \end{aligned} \tag{4.6}$$



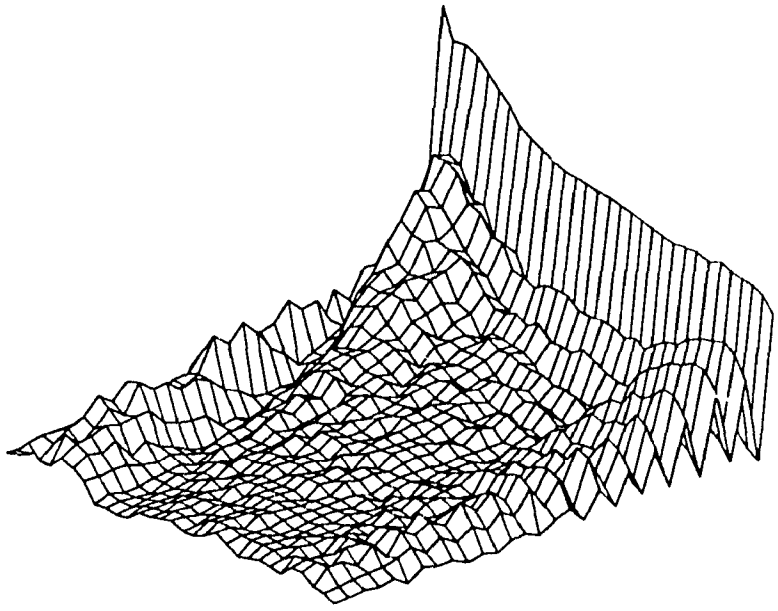
(a)



(b)



(c)



(d)

Figure 4.2: The intermediate stages when the feature extraction algorithm is applied to the character shown in Figure 4.1(a): its $|F(u,v)|$ (a), $G(\rho,\theta)$ (b), $|H(\rho,\phi)|$ (c), and the $T(\rho,\phi)$ obtained from the $|H(\rho,\phi)|$s of the set of multi-font Chinese characters used in the previous two chapters (d).

as being referred to in Eq.(4.3). Unfortunately, this is not the case in digital image processing, hence some subtleties in the transformation of discretized coordinate systems need to be addressed.

4.3.1 Resolution of Signal Images

If the coordinates are continuous, a one-to-one mapping can be established between the *Cartesian* coordinates and the polar coordinates. This, however, is untenable between the discrete coordinate systems. Presume the intervals between two consecutive discretized coordinate values in *Cartesian* coordinate system (u, v) and polar coordinate system (ρ, θ) are $(\Delta u, \Delta v)$ and $(\Delta\rho, \Delta\theta)$ respectively. Then their image pixels are $\Delta u \Delta v$ and $\rho_k \Delta\rho\Delta\theta$, as shown by the shaded areas in Figure 4.3, where $\rho_k = (k + \frac{1}{2})\Delta\rho$ $(k = 0, 1, 2, \cdots)$. Evidently, due to their resolutions, the area of a pixel in one coordinate system may correspond to part or the whole regions of more than one pixel of the other coordinate system, and *vice versa*.

One of the intuitive way to solve this problem with a predetermined precision requirement is to divide the image pixel into a group of smaller areas called subpixels. In this case, when *Cartesian* coordinates (u, v) are to be transformed into polar coordinates (ρ, θ), image pixel $\Delta u \Delta v$ will be divided into p^2 subpixels, so that each of them is small enough to be mapped into a unique pixel $\rho_k \Delta\rho\Delta\theta$ of polar coordinates, and the errors caused thereof is negligible. Therefore, instead of the original pixel with coordinates:

$$(u_i, v_j) = (i\Delta u, j\Delta v) \tag{4.7}$$

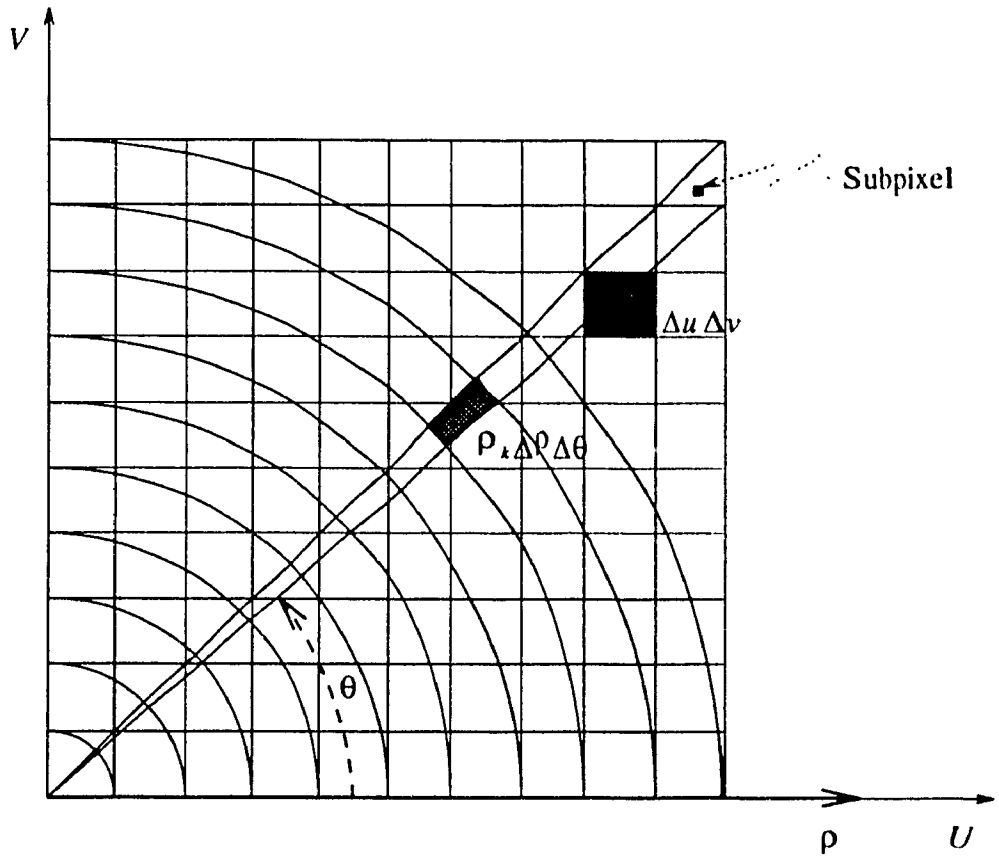


Figure 4.3: The discrete *Cartesian* coordinate system and polar coordinate system.

these subpixels will be referred to in the change of coordinate systems with coordinates:

$$(u_i + i_p\frac{\Delta u}{p}, v_j + j_p\frac{\Delta v}{p}) = ((i + \frac{i_p}{p})\Delta u, (j + \frac{j_p}{p})\Delta v) \tag{4.8}$$

where $i_p, j_p = 0, 1, \cdots, p-1$.

4.3.2 Image Sampling under Discrete Polar Coordinate System

After a mapping rule has been established between the two coordinate systems, what remains to be decided is the function value on each pixel $\rho_k \Delta\rho\Delta\theta$ of the polar coordinate system after the transformation. This is a bit more troublesome since it is actually a problem related to the manner of image sampling under polar coordinates.

Suppose $\tilde{f}(\tilde{u}, \tilde{v})$ is a two dimensional function with continuous *Cartesian* coordinates $(\tilde{u}, \tilde{v})$. Then its equally spaced sampling can be conducted as follows:

$$f(u_i, v_j) = \tilde{f}(u_i, v_j) = \tilde{f}(i\Delta u, j\Delta v) \tag{4.9}$$

or:

$$f(u_i, v_j) = \frac{1}{\Delta u \Delta v}\iint_{\Delta u \Delta v} \tilde{f}(\tilde{u}, \tilde{v})\, d\tilde{u}\, d\tilde{v} \tag{4.10}$$

which is a form of averaging over image pixels.

In both of the above cases, the area size of image pixels does not enter into the picture of the sampling functions. This is because the usually adopted equally spaced sampling happens to generate pixels with equal sizes in area under the *Cartesian* coordinate system. However, in a discrete polar system, this is no longer the case.

Now a pixel's area, which is:

$$\rho_k \Delta\rho\Delta\theta = (k + \frac{1}{2})\Delta\rho^2\Delta\theta \qquad k = 0, 1, 2, \cdots \tag{4.11}$$

is proportional to its distance from the origin. Therefore, during the discretization of polar coordinates, the information of not only the original function value on each pixel, but also its area size, should be expressed in the sampling function. One intuitive way is to take their product. That is, the sampling of function $\tilde{f}(\tilde{\rho}, \tilde{\theta})$ can be conducted as:

$$\begin{aligned} f(\rho_k, \theta_l) &= \rho_k \Delta\rho\Delta\theta \left(\frac{1}{\rho_k \Delta\rho\Delta\theta} \iint_{\rho_k \Delta\rho\Delta\theta} \tilde{f}(\tilde{\rho}, \tilde{\theta})\, \tilde{\rho}\, d\tilde{\rho}\, d\tilde{\theta} \right) \\ &= \iint_{\rho_k \Delta\rho\Delta\theta} \tilde{f}(\tilde{\rho}, \tilde{\theta})\, \tilde{\rho}\, d\tilde{\rho}\, d\tilde{\theta} \end{aligned} \tag{4.12}$$

With this sampling principle in mind, we now look into the details of the transformation process of the coordinate systems. Assume $f(u_i, v_j)$ and $g(\rho_k, \theta_l)$ are the functions of the same image under *Cartesian* and polar coordinates respectively. Since these two coordinate systems have already been discretized, the integration in Eqs.(4.10) and (4.12) is degraded into summation. Besides, in accordance with the sampling form of $g(\rho_k, \theta_l)$, $f(u_i, v_j)$ should also be treated as the product of the function and area of the pixel at (u_i, v_j). This is feasible since first of all the pixels here have the same area $\Delta u \Delta v$, and secondly, only the relative size of the pixels in the polar coordinate system is of significance, as can be seen from Eq.(4.11). Hence, the area of $\Delta u \Delta v$ can be assigned as 1 for simplicity, and the function value of each subpixel of the pixel at (u_i, v_j) becomes:

$$f(u_i + i_p \frac{\Delta u}{p}, v_j + j_p \frac{\Delta v}{p}) = \frac{1}{p^2} f(u_i, v_j) \qquad i_p, j_p = 0, 1, \cdots, p-1 \tag{4.13}$$

and also:

$$f(u_i, v_j) = \sum_{i_p=0}^{p-1} \sum_{j_p=0}^{p-1} f(u_i + i_p \frac{\Delta u}{p}, v_j + j_p \frac{\Delta v}{p}) = p^2 f(u_i + i_p \frac{\Delta u}{p}, v_j + j_p \frac{\Delta v}{p}) \quad (4.14)$$

Therefore, the transformed image function $g(\rho_k, \theta_l)$ becomes:

$$g(\rho_k, \theta_l) = \sum_{(u',v') \in \Omega} f(u', v') \quad (4.15)$$

where

$$\Omega = \{(u', v') | \; u' = u_i + i_p \frac{\Delta u}{p}, \; v' = v_j + j_p \frac{\Delta v}{p}, \; \rho_k \leq \sqrt{u'^2 + v'^2} < \rho_{k+1}, \; \theta_l \leq \arctan \frac{v'}{u'} < \theta_{l+1} \}.$$

4.4 Distinctiveness of the FPFB Extracted Feature Vectors

4.4.1 Binarization and the Vectors' Distinctiveness

The binarization in the last step of FPFB algorithm may turn the format of the extracted features into the one fit for the AMC network developed in our previous studies, as well as increase the robustness of the whole process against accumulated computational errors. This is because rather than the numerical value itself, such a step together with the threshold function generation algorithm has made only the feature signal's relative magnitude compared with that of the threshold function on each site be of interest, which is acquired on the basis of the whole group of patterns considered from a statistical point of view. That is, the threshold $T(\rho, \phi)$ is obtained

first through a stocastic process based on all the $|H_k(\rho,\phi)|$s ($k = 1, 2, \cdots, s$). Then each $|H_k(\rho,\phi)|$ is compared with $T(\rho,\phi)$ by a subtraction operation, and only the sign but the value of the result is retained.

The format change of feature vectors as well as the removal of the accumulated computational errors are achieved through binarization by discarding irrelevant details originally contained in feature vectors. Only the shape of each feature pattern relative to all the rest among the same group is of importance. Different feature patterns with different shapes should be binarized into different binary or bipolar vectors, namely, the primitive distinctiveness among the extracted feature patterns should be retained after binarization. Nonetheless, binarization is carried out on a threshold function generated through statistics of the whole group of feature patterns. If there exist two or more pattern signals which lie most frequently on the same side of the threshold function, although the original shapes of these signals may be quite different, their binarized vectors will be very similar.

In fact, compared with that of the threshold function, this situation may come into being only when the energy of all these signals is obviously larger or smaller. Assume the energy of all these signals is about the same, so is that of the threshold function. As a result, if a certain component of feature signal is larger than the threshold there, then there should be somewhere else that it is smaller. In this case, any two feature vectors are not likely to be similar after binarization provided they are primitively very different in shape. Therefore, to eliminate such a possibility, the energy equivalent mechanism could be introduced into our feature extraction process.

4.4.2 Signal's Energy Equivalence Mechanism in Feature Extraction

To acquire the energy equivalent signal through the FPFB featrue extraction process, a few modifications of Algorithm 2 is needed. The newly devised algorithm will be called EEFPFB algorithm, in which "EE" stands for energy equivalent.

Algorithm 4 *EEFPFB Algorithm:*

1. *For a specific character image $b(x,y)$ defined as in Eq.(4.1) of Section 4.2.1, count the number of its 1-valued pixels as k;*

2. *Scale the function $b(x,y)$ by $\frac{1}{\sqrt{k}}$, which gives a real-numbered function $f(x,y)$:*

$$f(x,y) = \frac{1}{\sqrt{k}} b(x,y)$$

3. *Conduct FPFB Algorithm with $f(x,y)$ as its input, but in its second step (See Algorithm 2), make the replacement of coordinate systems with $f(x,y)$'s power spectrum so that the newly acquired spectrum $G(\rho,\theta)$ satisfies:*

$$\begin{aligned} G^2(\rho,\theta) &= |F(u(\rho,\theta), v(\rho,\theta))|^2 \\ &= |F(\rho\cos\theta, \rho\sin\theta)|^2 \end{aligned} \tag{4.16}$$

Based on this design, the signal energy equivalent nature of EEFPFB algorithm can be revealed by the following proposition:

Theorem 6 *Assume $b_1(x,y)$ and $b_2(x,y)$ are two different character images with k_1 and k_2 1-valued pixels respectively. Then during the application of EEFPFB al-*

gorithm, $|H_1(\rho,\phi)|$ and $|H_2(\rho,\phi)|$, their corresponding signals obtained through the series of transformations right before binarization, are equivalent in their energy.

Proof:

The energy of the original character image $b_1(x,y)$ is:

$$\sum b_1^2(x,y) = \underbrace{1+1+\cdots+1}_{k_1} = k_1$$

So:

$$\begin{aligned}\sum f_1^2(x,y) &= \sum [\frac{1}{\sqrt{k_1}} b_1(x,y)]^2 \\ &= \frac{1}{k_1}\sum b_1^2(x,y) \\ &= 1\end{aligned}$$

This is also true for character image $b_2(x,y)$. Hence:

$$\sum f_1^2(x,y) = \sum f_2^2(x,y) \tag{4.17}$$

That is, the first two steps of EEFPFB algorithm is actually an energy equalization process of character images.

Now according to Parseval's theorem, the energy of signal is conservative in Fourier transforms. Therefore, after the transformations in steps one and three of FPFB algorithm (See Section 4.2.1), the signals, taken character image $b_1(x,y)$ as an example, satisfy the following relations separatively:

$$\begin{aligned}\sum |F_1(u,v)|^2 &= \sum f_1^2(x,y) \\ \sum |H_1(\rho,\phi)|^2 &= \sum G_1^2(\rho,\theta)\end{aligned}$$

So, by Eq (4.16):

$$\begin{aligned}\sum |H_1(\rho,\phi)|^2 &= \sum G_1^2(\rho,\theta) \\ &= \sum |F_1(u,v)|^2 \\ &= \sum f_1^2(x,y)\end{aligned}$$

Similarly:

$$\sum |H_2(\rho,\phi)|^2 = \sum f_2^2(x,y)$$

Therefore, according to Eq.(4.17):

$$\sum |H_1(\rho,\phi)|^2 = \sum |H_2(\rho,\phi)|^2$$

That completes the proof of this theorem.

4.4.3 Applicability of EEFPFB Algorithm

Binarization is always fulfilled by discarding some detailed information. Therefore, it is natural to concern about the possibility for the patterns to lose their distinctiveness through binarization. Theoretically, the energy equivalent mechanism of pattern signals has the capability to reduce such a possible loss due to the signal energy related reasons. But in practice, whether it is worth being put into use depends on how these energy related factors will affect the distinctiveness of vectors after binarization. Different sets of data may respond to this differently. However, whatever happens, the ultimate purpose of our feature extraction algorithms is to make the feature patterns suitable to the associative memory classifier in addition to the detection of certain desired deformation invariant features. In Section 2.4.3, two means for measuring

the suitability of data's fitness to AMC have been set up, one of which is the average correlation coefficient of the set of vectors. If a set of feature vectors generated from a feature extraction algorithm is more suitable to AMC, then this algorithm is more desirable. For this reason, these measurements may also be used to inspect the performance of a feature extraction process.

The average correlation coefficients of feature vectors extracted from the set of 255 multi-font Chinese characters utilized before by both EEFPFB and FPFB algorithms are displayed in Table 4.1, together with their absolute difference. Under each of these two algorithms, the threshold function is constructed from the whole set of 255 feature vectors. These average correlation coefficients are listed first in agreement with the three subgroups of vectors in regard to every character font, and then the entire set of data. To see how far the similarity of any two vectors in each case can reach, the maximal correlation coefficients are also tabulated. In Eq.(2.17), the correlation coefficient is defined in the sense of absolute value. This is because for any two m-dimensional bipolar vectors $\vec{x}_k$ and $\vec{x}_l$, there exists:

$$-1 \leq \frac{\vec{x}_k \vec{x}_l^T}{m} \leq 1$$

and only the absolute value reflects the degree of their similarity, its sign however, is nothing but an indication of their relative directions, i.e. when any one of them is projected to the other, whether it will on the same or the oppisite direction of the latter. Therefore, to show the range within which $\frac{\vec{x}_k \vec{x}_l^T}{m}$ may vary, here the maximal correlation coefficients are displayed separately in accordance with the original sign of $\frac{\vec{x}_k \vec{x}_l^T}{m}$.

		EEFPFB	FPFB	Absolute Difference
Song	positive maximum	0.4090	0.3673	– – –
	negative maximum	−0.2175	−0.2300	– – –
	average	0 0821	0.0722	0.0099
Kai	positive maximum	0.2841	0.3652	– – –
	negative maximum	−0.1800	−0.1717	– – –
	average	0.0692	0.0718	0.0026
Boldface	positive maximum	0.5463	0.4651	– – –
	negative maximum	−0.2300	−0.1488	– – –
	average	0.2466	0.1193	0.1273
Entire Set	positive maximum	0.5463	0.4651	– – –
	negative maximum	−0.3757	−0.2882	– – –
	average	0.0981	0.0698	0.0283

Table 4.1: The average correlation coefficients of the EEFPFB and FPFB extracted feature vectors and their absolute difference together with the maxima obtained on the basis of their original sign, listed in accordance with each font and the entire data set. The binarization is conducted on the same threshold functions generated from the entire data set

The statistics in Table 4.1 suggests that it is impossible for two or more different patterns to share one and the same bipolar vector after binarization at least in the current FPFB-type algorithms. The maxima ever reached are around 0.5 under these circumstances, and the averages over the entire data set are below 0.1. In addition, this example even implies that it is unnecessary to concern about the loss of pattern's distinctiveness due to the energy of their signals in a set of real-life data. It is expected that the extracted feature vectors will be more distinct from one another if the pattern signals used in the construction of threshold function are distributed more randomly. Yet, the application of energy equalization introduces additional regularity into these pattern signals, and under certain circumstances this may cause problem. For example, in our present case, the whole set of original data contains character images of three different fonts. The energy of a character image depends on the number of character pixels it has. Different fonts are shaped from the written styles of a character's strokes. Some are thick, others are thin. Characters in a font of a thicker stroke will have more pixels and therefore higher energy. In our data, as shown in Figure 2.7(a), characters in font boldface possess more pixels, while those in the other two fonts seem similar in this respect. For this reason, energy equalization has introduced extra similarity among characters of the same font, as can be seen in Table 4.1. This effect is especially severe in font boldface, and on the whole, makes the average correlation coefficient over the entire set of vectors extracted by EEFPFB a bit more larger than that by the FPFB algorithm.

This decline in feature vectors' distinctiveness is checked specially to see whether

it is really caused by grouped regularity in energy distribution of the primitive data. The results are shown in Table 4.2. This time the threshold functions are generated separately from each subset of characters of a single font. Now that the interference of different fonts in energy distribution has been ruled out, there is almost no difference between the extraction results of EEFPFB and FPFB algorithm in terms of average correlation coefficient in all the three fonts, as is indicated by their absolute differences. Besides, the varying range of $\frac{\vec{x}_k \vec{x}_l^T}{m}$ bounded by both positive and negative maximum on either of its two ends is shifted to be more symmetric with regard to 0 in each of these fonts compared with their corresponding cases in Table 4.1, which means the reduction of similarity among feature vectors extracted from the characters of the same font.

Based on the above discussion, our studies about the feacure extraction process below will mainly be focused on the application of FPFB algorithm.

4.3 Improvement in Pattern Vectors Fitness to AMC through the FPFB Algorithm

In our previous studies on the development of AMC, a set of 85 Chinese characters each in three fonts are utilized for simulation purpose. So totally there are 255 different character images in use. The input patterns are formed by simply lining up rows of pixels of the character images into one dimensional vectors, called original character vectors.

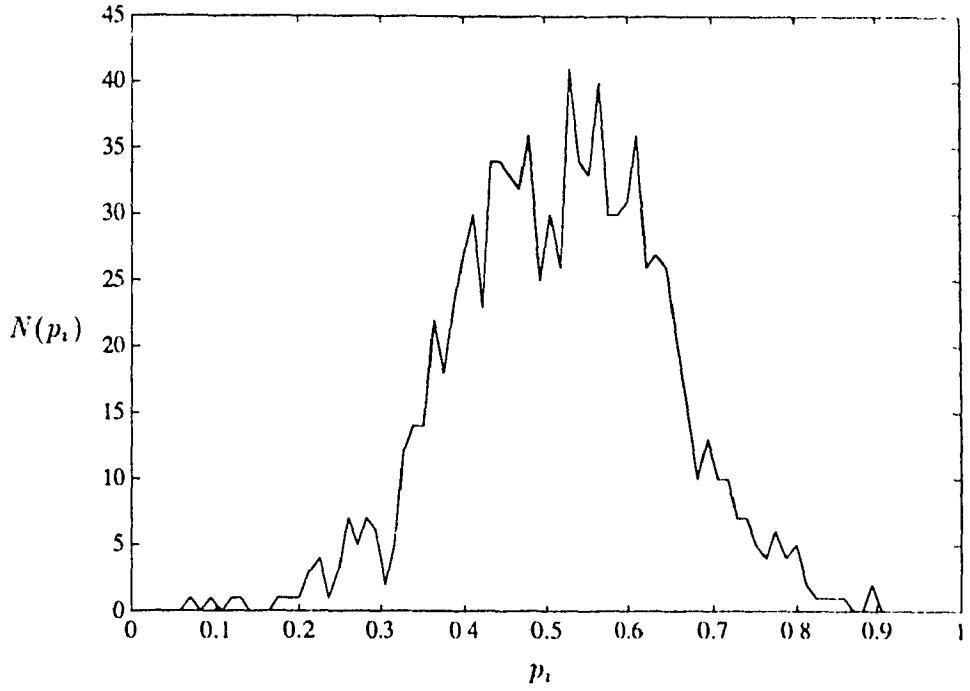
		EEFPFB	FPFB	Absolute Difference
Song	positive maximum	0.3382	0.3195	– – –
	negative maximum	−0.2778	−0.2383	– – –
	average	0.0609	0.0596	0.0013
Kai	positive maximum	0.2737	0.2924	– – –
	negative maximum	−0.2196	−0.2133	– – –
	average	0.0532	0.0527	0.0005
Boldface	positive maximum	0.3007	0.3569	– – –
	negative maximum	−0.2737	−0.2612	– – –
	average	0.0654	0.0659	0.0005

Table 4.2: The average correlation coefficients of the EEFPFB and FPFB extracted feature vectors and their absolute difference together with the maxima obtained on the basis of their original sign, listed in accordance with each font. The binarization is conducted on the different threshold functions generated from vector subsets of each kind of character font.

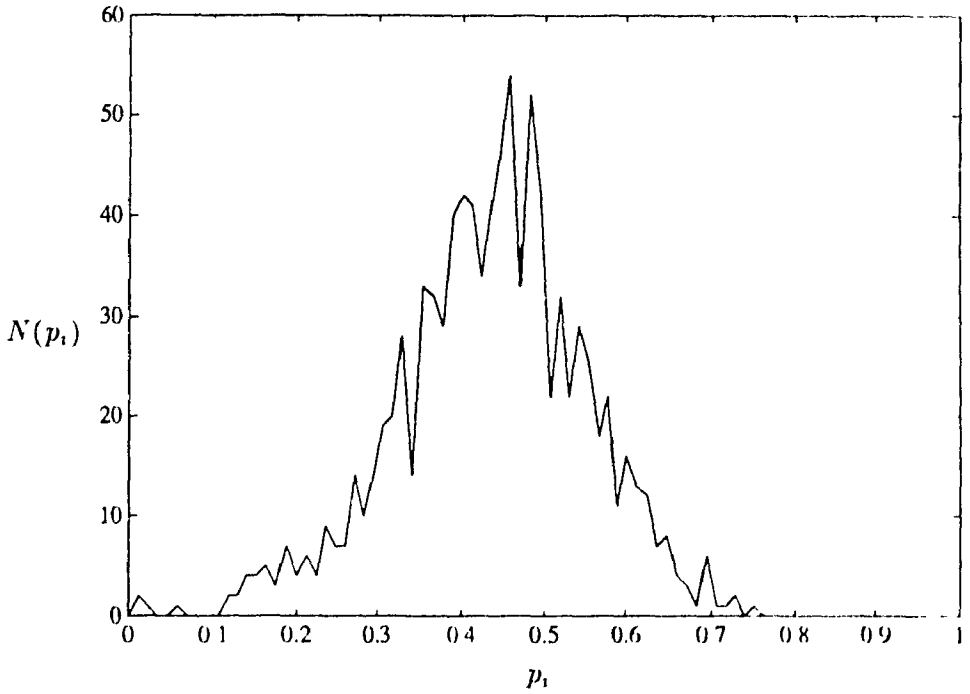
Data Group	Song	Kai	Boldface	The Entire Set
Correlation Coefficient of the Original Character Vectors	0.3002	0.3472	0.1580	0.2570
Correlation Coefficient of the FPFB Extracted Feature Vectors	0.0722	0.0718	0.1193	0.0698

Table 4.3: The average correlation coefficients of the original character vectors and the FPFB extracted feature vectors, listed in accordance with each font and the entire data set.

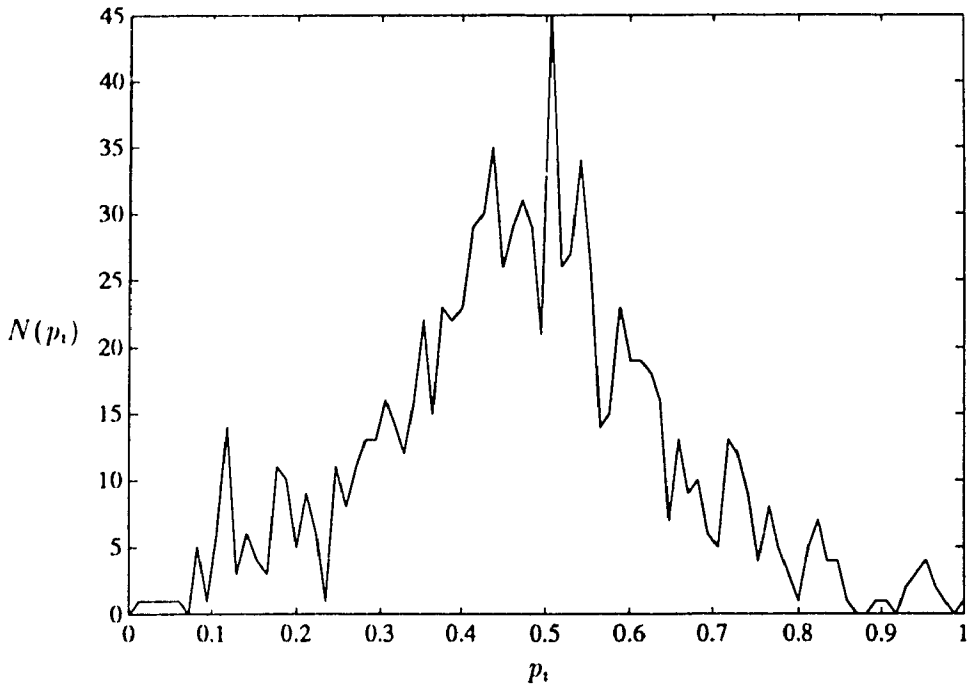
To demonstrate the ability of the FPFB algorithm in improving the suitability of the pattern vectors to AMC, both the input patterns made of original character vectors and those obtained by applying the FPFB algorithm to this same group of character data are examined with the measurements established in Section 2.4.3. The average correlation coefficients of these two groups of input vectors are tabulated in Table 4.3 for each font of 85 Chinese characters and the entire set of 255 character images. Also, the components' distributions on their probability $p_i = P\{x_i = 1\}$ are illustrated in Figures 4.4 for the FPFB extracted feature vectors. The abscissas p_i are a component's probability $p_i = P\{x_i = 1\}$, and the ordinates $N(p_i)$ represent the number of such components. Those for the original character vectors have been displayed in Figure 2.6.


45
40
35
30
25
20
15
10
5
0
$N(p_i)$
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
p_i

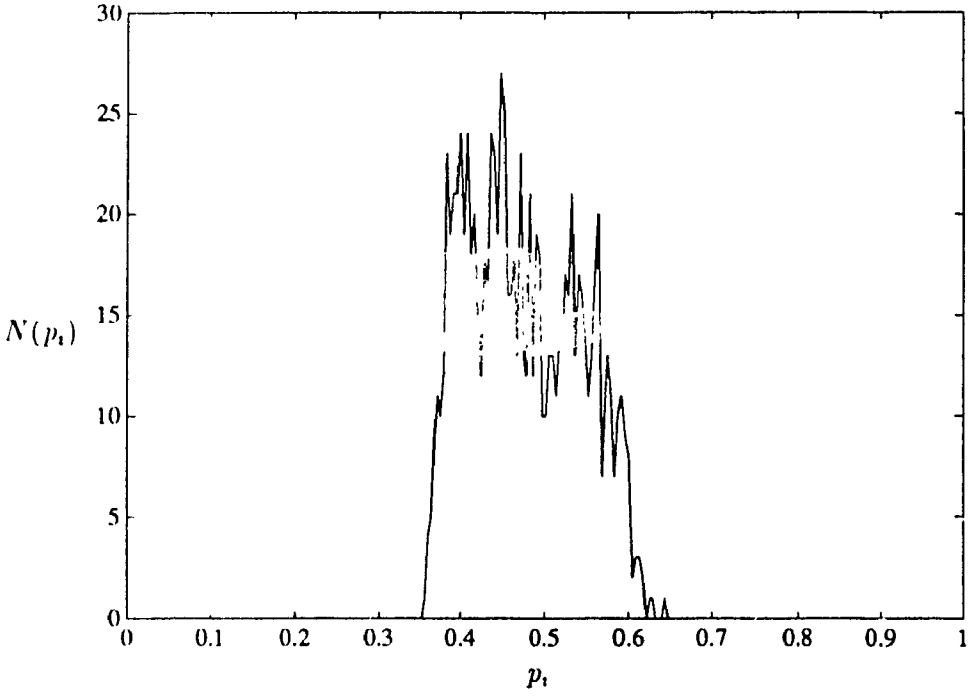
(a)


60
50
40
30
20
10
0
$N(p_i)$
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
p_i

(b)



(c)



(d)

Figure 1.4: 'Components' distributions on their probability $p_i = P\{x_i = 1\}$ of the FPFB extracted feature vectors, listed on the basis of each font, *i.e.* Song (a), Kai (b), and boldface (c) respectively, and the entire data set (d).

According to its definition, the average correlation coefficient μ varies within the range of 0 to 1. Since a set of pattern vectors is expected to be more orthogonal to one another as far as possible, in the ideal case, μ should approach 0. Therefore the effect of the binarization step together with the threshold function generation algorithm of the FPFB feature extraction process is quite evident.

In a set of real-life data, the perfection in the identical componental probability distribution assumption of Eq.(2.27) is untenable. Under this circumstance, it is expected that all the p_is may concentrate on a certain value as far as they can to approximate such a condition. In this sense, part (d) of Figures 2.6 and 4.4 shows a significant improvement in the fitness of the input vectors to AMC through FPFB transformation. This change in parts (a) (b) and (c) of these two figures which depict the case of each font is not as obvious since the generation of threshold function is conducted only on the entire data set. However, according to Eq.(2.32), $\vec{x}_k \vec{x}_l^T = 0$ if $p = \frac{1}{2}$. Therefore when each single font is examined, the improvement after applying the FPFB algorithm can also be perceived for all the distributions of p_is in (a) through (c) of Figures 4.4 tend to be shifted to around $p_i = \frac{1}{2}$.

4.6 Simulation on Character Recognition

It is common knowledge that the ultimate usefulness of a feature extraction process depends on the performance of a pattern recognition system in which it has been incorporated. In this section, the effectiveness of the FPFB algorithm will be tested by the recognition of several sets of Chinese characters with a system composed of

this feature extraction algorithm and an associative memory classifier.

The input patterns to the AMC are vectors extracted by FPFB from the character images to be recognized, like that shown in Figure 4.1(b). Their corresponding associated output patterns are still the set of Hadamard vectors. Once again, the set of recognition criteria laid down in Section 2.6.1 on the largest similarity value (LSV) of the recollected vector and the Hadamard vectors is used.

In our experiments below, two groups of data will be involved.

4.6.1 On the set of Multi-font Chinese characters Used Formerly

The first group of data tested is the set of multi-font Chinese characters used in the previous two chapters. It contains the first 85 characters in our data base of 3114 printed ones each in three different fonts, *i.e.* Song, Kai, and boldface. So a total of 255 different character images is involved.

The shift of original pattern by pixels does not cause any difference to its Fourier spectrum unless the character is truncated due to translation. So the major concern in our experiments is the system's performance on rotative distortions. Nine groups of data, which are rotated by 0°, 5°, 15°, 30°, 45°, 60°, 90°, 180°, and 270° respectively, are utilized. Their recognition results are displayed in Table 4.4, in terms of the numbers of misclassification (above) and rejection (below) occurred in each font and the entire data set, together with the corresponding recognition, rejection, and error rates of the latter. Also the average LSV's of different testing data groups over each font and the entire set are listed in Table 4.5. Here, the pairing scheme adopted in

the construction of AMC is based on the order of the natural appearance of the input vectors.

To see the efficacy of the pairing strategy discussed in last chapter and in the end find a better inner coding scheme, both the group of testing data without rotation, *i.e.* rotated by 0°, and the group which has the lowest recognition rate in Table 4.4, namely the group of data rotated by 60° in the present case, will be tested under the pairing schemes when the input data set has been divided first into 1 (the case of the order of their natural appearance), 2, 4, through 64 subgroups respectively. These results are shown in Table 4.6 to Table 4.9 in the same format as either Table 4.4 or Table 4.5. The best pairing scheme occurred to the latter group of testing data in terms of recognition rate will be regarded as an estimation on how well the entire system can perform, and therefore all the other groups of testing data will be examined with it. Here specifically, the best is the one when pairing takes place after the division of the input pattern data first into 16 subgroups. The results of this experiment are illustrated by both Table 4.10 and Table 4.11.

The recognition rates for all groups of testing data are above 96% in Table 4.4, while in Table 4.10 all misclassifications have been eliminated, and the worst are the cases where there is only one rejection. This convincingly demonstrates the effectiveness of the FPFB feature extraction algorithm against geometric distortions. For the group of characters without deformation, a completely correct classification has been achieved. Since no additional errors other than the rotation itself are introduced when characters are tilted upside down or fallen down side-ways, the outcomes for those

	Testing Data Group								
	0°	5°	15°	30°	45°	60°	90°	180°	270°
Song	0	0	0	0	0	0	0	0	0
	0	0	0	1	0	2	0	0	0
Kai	0	0	0	0	0	0	0	0	0
	0	0	0	0	3	1	0	0	0
Boldface	0	1	0	2	2	6	0	0	0
	0	1	1	3	0	0	0	0	0
Entire Set	0	1	0	2	2	6	0	0	0
	0	1	1	4	3	3	0	0	0
Recognition Rate	100.0%	99.22%	99.61%	97.65%	98.04%	96.47%	100.0%	100.0%	100.0%
Rejection Rate	0.0%	0.39%	0.39%	1.57%	1.18%	1.18%	0.0%	0.0%	0.0%
Error Rate	0.0%	0.39%	0.0%	0.78%	0.78%	2.35%	0.0%	0.0%	0.0%

Table 4.4: The numbers of misclassification and rejection in different testing data groups, listed in line with each font and the entire data set, together with the corresponding recognition, rejection, and error rates of the latter for the set of multi-font Chinese characters used before. The pairing is conducted under the order of the natural appearance of the input vectors.

	Testing Data Group								
	0°	5°	15°	30°	45°	60°	90°	180°	270°
Song	197.56	169.87	170.76	170.20	171.08	170.09	197.56	197.56	197.56
Kai	200.44	167.38	167.98	166.19	167.13	166.11	200.45	200.44	200.45
Boldface	192.16	169.33	169.31	166.88	168.40	166.49	192.16	192.16	192.16
Entire Set	196.72	168.86	169.35	167.76	168.87	167.56	196.73	196.72	196.73

Table 4.5: The average largest similarity value (LSV) obtained during the associative recollection processes with different testing data groups for the recognition of the set of multi-font Chinese characters used before. The results are listed on the basis of each font and the entire data set. The average LSV's may vary from 0 to 256. And pairing is conducted based on the order of the natural appearance of the input vectors.

	Grouping Status (The No. of Subgroups)						
	1	2	4	8	16	32	64
Song	0	0	0	0	0	0	0
	0	0	0	0	0	0	0
Kai	0	0	0	0	0	0	0
	0	0	0	0	0	0	0
Boldface	0	0	0	0	0	0	0
	0	0	0	0	0	0	0
Entire Set	0	0	0	0	0	0	0
	0	0	0	0	0	0	0
Recognition Rate	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
Rejection Rate	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Error Rate	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

Table 4.6: The numbers of misclassification and rejection in the group of testing data without rotation, listed in line with each font and the entire data set, together with the corresponding recognition, rejection, and error rates of the latter for the set of multi-font Chinese characters used before. The pairing is conducted after the input data set has been divided first into 1, 2, 4, ..., through 64 subgroups separately.

	Grouping Status (The No. of Subgroups)						
	1	2	4	8	16	32	64
Song	197.56	205.69	207.04	208.66	209 04	209.25	209.56
Kai	200.44	206.75	209.13	209.79	210.44	210.96	211.14
Boldface	192.16	200.94	200.93	202.29	203.36	203.72	203.36
Entire Set	196.72	204.46	205.70	206.91	207.61	207.98	208.02

Table 4.7: The average largest similarity value (LSV) obtained during the associative recollection processes with the group of testing data without rotation for the recognition of the set of multi-font Chinese characters used before. The results are listed on the basis of each font and the entire data set. The average LSVs may vary from 0 to 256. And pairing is conducted after the input data set has been divided first into 1, 2, 4, ..., through 64 subgroups separately.

	Grouping Status (The No. of Subgroups)						
	1	2	4	8	16	32	64
Song	0	0	0	0	0	0	0
	2	0	0	0	0	0	0
Kai	0	2	0	0	0	1	0
	1	0	1	0	0	0	0
Boldface	6	2	0	1	0	0	1
	0	0	0	0	0	1	0
Entire Set	6	4	0	1	0	1	1
	3	0	1	0	0	1	0
Recognition Rate	96.47%	98.43%	99.61%	99.61%	100.0%	99.22%	99.61%
Rejection Rate	1.18%	0.0%	0.39%	0.0%	0.0%	0.39%	0.0%
Error Rate	2.35%	1.57%	0.0%	0.39%	0.0%	0.39%	0.39%

Table 4.8: The numbers of misclassification and rejection in the group of testing data rotated by 60°, listed in line with each font and the entire data set, together with the corresponding recognition, rejection, and error rates of the latter for the set of multi-font Chinese characters used before. The pairing is conducted after the input data set has been divided first into 1, 2, 4, ..., through 64 subgroups separately.

	Grouping Status (The No. of Subgroups)						
	1	2	4	8	16	32	64
Song	170.09	173.92	174.18	174.14	175.44	176.08	175.60
Kai	166.11	168.64	170.65	172.33	172.86	172.49	172.51
Boldface	166.49	170.59	171.35	171.58	171.67	171.74	171.76
Entire Set	167.56	171.05	172.06	172.68	173.32	173.44	173.29

Table 4.9: The average largest similarity value (LSV) obtained during the associative recollection processes with the group of testing data rotated by 60° for the recognition of the set of multi-font Chinese characters used before. The results are listed on the basis of each font and the entire data set. The average LSVs may vary from 0 to 256. And pairing is conducted after the input data set has been divided first into 1, 2, 4, ..., through 64 subgroups separately.

	Testing Data Group								
	0°	5°	15°	30°	45°	60°	90°	180°	270°
Song	0	0	0	0	0	0	0	0	0
	0	0	1	0	0	0	0	0	0
Kai	0	0	0	0	0	0	0	0	0
	0	0	0	0	1	0	0	0	0
Boldface	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0
Entire Set	0	0	0	0	0	0	0	0	0
	0	0	1	0	1	0	0	0	0
Recognition Rate	100.0%	100.0%	99.61%	100.0%	99.61%	100.0%	100.0%	100.0%	100.0%
Rejection Rate	0.0%	0.0%	0.39%	0.0%	0.39%	0.0%	0.0%	0.0%	0.0%
Error Rate	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

Table 4.10: The numbers of misclassification and rejection in different testing data groups, listed in line with each font and the entire data set, together with the corresponding recognition, rejection, and error rates of the latter for the set of multi-font Chinese characters used before. The pairing is conducted after the input data set has been divided first into 16 subgroups.

	Testing Data Group								
	0°	5°	15°	30°	45°	60°	90°	180°	270°
Song	209.04	175.19	176.93	174.86	177.68	175.44	209.04	209.04	209.04
Kai	210.44	173.20	175.18	172.95	175.34	172.86	210.46	210.44	210.46
Boldface	203.36	172.66	172.49	172.16	173.74	171.67	203.36	203.36	203.36
Entire Set	207.61	173.68	174.87	173.33	175.59	173.32	207.62	207.61	207.62

Table 4.11: The average largest similarity value (LSV) obtained during the associative recollection processes with different testing data groups for the recognition of the set of multi-font Chinese characters used before. The results are listed on the basis of each font and the entire data set. The average LSVs may vary from 0 to 256. And pairing is conducted after the input data set has been divided first into 16 subgroups.

groups of data rotated by a multiple of 90° are the same as those for the unrotated patterns. If the character patterns are rotated with an angle other than these, due to the discretization of the coordinates, some errors are introduced. This may causes severe errors in the series of transformations, especially in the high frequency part, and in the end leads to a possible rejection or misclassification, or at least a reduction in the average LSV, the confidence indication of the classification process, as can be seen in Tables 4.5 and 4.11.

The improvement in the suitability of the feature vectors extracted from the FPFB algorithm for the AMC can be seen clearly here through comparisons between the results listed in Tables 3.1 and 4.6, and Tables 3.2 and 4.7 respectively. With the FPFB extracted feature vectors as input patterns, not only all the character images can be recognized correctly if there is no distortions occurred to them, but also the average LSVs acquired during associative recollection are increased significantly. In addition, the increments in the average LSV before and after the implementation of the pairing strategy developed in Chapter 3 are more than doubled. All these are resulted directly from such an improvement of these feature vectors compared with the data used formerly.

4.6.2 On a Set of Multi-font Chinese Characters of Similar Character Subsets

Associative memory classifier is developed originally in the hope of solving the recognition problem of large number categories. The testing data set used in our experiment

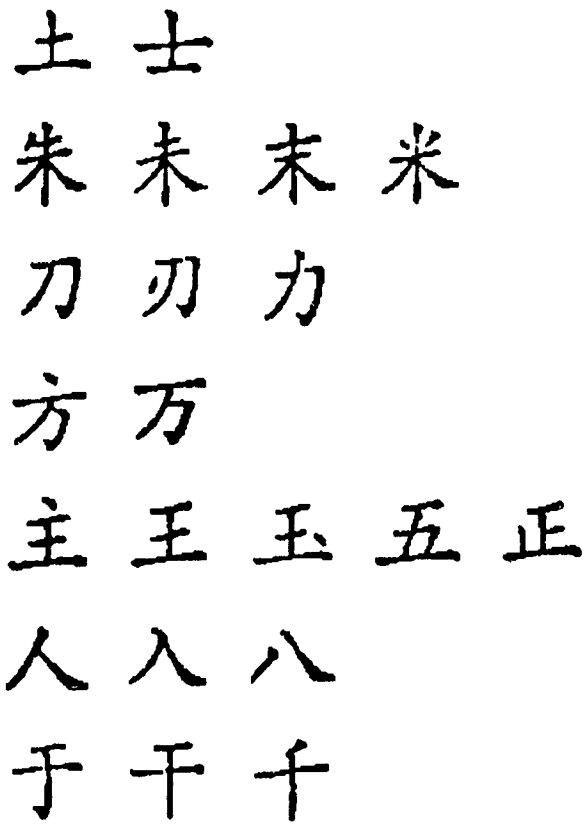


Figure 4.5: Some subsets of similar Chinese characters used in simulation.

has already reached the scale of hundreds of categories. However, recognition problems like that of Chinese characters may have pattern classes counted by thousands. Yet instead of single layer ones, such problems are usually tackled with hierarchical classifiers, as has been indicated in Section 1.2.2. Therefore, the AMCs of our present scale may have a good chance to be utilized as parts of a multi-level classifier. Under these circumstances, more likely they will face patterns of greater similarities.

To cope with such a situation, our currently developed character recognition system is used to recognize another set of data composed of similar Chinese characters. This set of data contains 34 subsets of characters which look very much alike. Some

subsets of these characters are illustrated in Figure 4.5 (The entire set of data is shown in Appendix B). The total number of different characters is 85, and each character has three fonts as before. So there are still 255 character images involved totally. The testing data are formed by introducing the same sorts of distortions as those for the first set of data. Also, the experiments are conducted in the similar manner as that with the previous set of data. But this time the worst case happens to the group of data rotated by 30°, and the best pairing scheme is obtained by dividing first the feature patterns of this set of characters into 8 subgroups. All the results for these are displayed in Tables 4.12 through 4.19 in the same format as Tables 4.4 to 4.11.

Again, when there is no distortion to the system inputs, all the patterns will be recognized correctly. Besides, only very small amounts of degradation are observed when comparisons are made between the corresponding data in Tables 4.4 and 4.12, and Tables 4.5 and 4.13, especially in terms of average LSV. Such differences are even slighter under the conditions of selected pairing schemes, as shown in Tables 4.11 and 4.19. Comprehensibly, the similarities among characters in this set of data may cause these declines. However, the differences are so slight that it can be claimed that such a system is definitely stable with regard to the data to be processed. This, in large part, is also due to the effectiveness of the FPFB feature extraction algorithm.

	Testing Data Group								
	0°	5°	15°	30°	45°	60°	90°	180°	270°
Song	0	0	0	2	0	0	0	0	0
	0	0	0	3	0	0	0	0	0
Kai	0	4	0	3	0	1	0	0	0
	0	0	2	1	1	1	0	0	0
Boldface	0	2	1	0	2	1	0	0	0
	0	1	1	1	1	2	0	0	0
Entire Set	0	6	1	5	2	2	0	0	0
	0	1	3	5	2	3	0	0	0
Recognition Rate	100.0%	97.25%	98.43%	96.08%	98.43%	98.04%	100.0%	100.0%	100.0%
Rejection Rate	0.0%	0.39%	1.18%	1.96%	0.78%	1.18%	0.0%	0.0%	0.0%
Error Rate	0.0%	2.35%	0.39%	1.96%	0.78%	0.78%	0.0%	0.0%	0.0%

Table 4.12: The numbers of misclassification and rejection in different testing data groups, listed in line with each font and the entire data set, together with the corresponding recognition, rejection, and error rates of the latter for a set of multi-font Chinese characters of similar character subsets. The pairing is conducted under the order of the natural appearance of the input vectors.

	Testing Data Group								
	0°	5°	15°	30°	45°	60°	90°	180°	270°
Song	194.81	166.72	168.06	167.05	169.13	167.75	194.81	194.81	194.81
Kai	194.54	165.05	166.00	164.76	165.79	163.92	194.54	194.54	194.54
Boldface	189.74	167.58	169.16	165.59	167.45	166.65	189.74	189.74	189.74
Entire Set	193.03	166.45	167.74	165.80	167.45	166.11	193.03	193.03	193.03

Table 4.13: The average largest similarity value (LSV) obtained during the associative recollection processes with different testing data groups for the recognition of a set of multi-font Chinese characters of similar character subsets. The results are listed on the basis of each font and the entire data set. The average LSVs may vary from 0 to 256. And pairing is conducted based on the order of the natural appearance of the input vectors.

	Grouping Status (The No. of Subgroups)						
	1	2	4	8	16	32	64
Song	0	0	0	0	0	0	0
	0	0	0	0	0	0	0
Kai	0	0	0	0	0	0	0
	0	0	0	0	0	0	0
Boldface	0	0	0	0	0	0	0
	0	0	0	0	0	0	0
Entire Set	0	0	0	0	0	0	0
	0	0	0	0	0	0	0
Recognition Rate	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
Rejection Rate	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Error Rate	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

Table 4.14: The numbers of misclassification and rejection in the group of testing data without rotation, listed in line with each font and the entire data set, together with the corresponding recognition, rejection, and error rates of the latter for a set of multi-font Chinese characters of similar character subsets. The pairing is conducted after the input data set has been divided first into 1, 2, 4, ..., through 64 subgroups separately.

	Grouping Status (The No. of Subgroups)						
	1	2	4	8	16	32	64
Song	194.81	204.35	207.34	208.16	209.52	209.56	209.69
Kai	194.51	202.67	207.39	208.27	209.32	209.12	209.68
Boldface	189.74	197.89	200.05	200.74	201.31	202.55	201.88
Entire Set	193.03	201.64	204.93	205.73	206.71	207.08	207.09

Table 4.15: The average largest similarity value (LSV) obtained during the associative recollection processes with the group of testing data without rotation for the recognition of the set of multi-font Chinese characters of similar character subsets. The results are listed on the basis of each font and the entire data set. The average LSVs may vary from 0 to 256. And pairing is conducted after the input data set has been divided first into 1, 2, 4, ..., through 64 subgroups separately.

	Grouping Status (The No. of Subgroups)						
	1	2	4	8	16	32	64
Song	2	0	0	0	2	2	0
	3	1	0	0	0	0	1
Kai	3	3	3	0	0	0	1
	1	1	1	1	1	0	0
Boldface	0	0	1	0	1	0	0
	.	0	0	0	0	0	0
Entire Set	5	3	4	0	3	2	1
	5	2	1	1	1	0	1
Recognition Rate	96.08%	98.04%	98.04%	99.61%	98.43%	99.22%	99.22%
Rejection Rate	1.96%	0.78%	0.39%	0.39%	0.39%	0.0%	0.39%
Error Rate	1.96%	1.18%	1.57%	0.0%	1.18%	0.78%	0.39%

Table 4.16: The numbers of misclassification and rejection in the group of testing data rotated by 30°, listed in line with each font and the entire data set, together with the corresponding recognition, rejection, and error rates of the latter for the set of multi-font Chinese characters of similar character subsets. The pairing is conducted after the input data set has been divided first into 1, 2, 4, ..., through 64 subgroups separately.

	Grouping Status (The No. of Subgroups)						
	1	2	4	8	16	32	64
Song	167.05	172.33	175.00	174.62	175.45	175.76	175.95
Kai	164.76	169.84	173.87	173.00	174.55	174.33	174.21
Boldface	165.59	169.58	170.82	171.41	170.53	171.82	171.98
Entire Set	165.80	170.58	173.23	173.01	173.51	173.97	174.05

Table 4.17: The average largest similarity value (LSV) obtained during the associative recollection processes with the group of testing data rotated by 30° for the recognition of the set of multi-font Chinese characters of similar character subsets. The results are listed on the basis of each font and the entire data set. The average LSVs may vary from 0 to 256. And pairing is conducted after the input data set has been divided first into 1, 2, 4, ..., through 64 subgroups separately.

	Testing Data Group								
	0°	5°	15°	30°	45°	60°	90°	180°	270°
Song	0	0	0	0	0	1	0	0	0
	0	0	1	0	0	0	0	0	0
Kai	0	0	0	0	1	0	0	0	0
	0	0	0	1	1	0	0	0	0
Boldface	0	0	0	0	0	0	0	0	0
	0	0	0	0	1	1	0	0	0
Entire Set	0	0	0	0	1	1	0	0	0
	0	0	1	1	2	1	0	0	0
Recognition Rate	100.0%	100.0%	99.61%	99.61%	98.82%	99.22%	100.0%	100.0%	100.0%
Rejection Rate	0.0%	0.0%	0.39%	0.39%	0.78%	0.39%	0.0%	0.0%	0.0%
Error Rate	0.0%	0.0%	0.0%	0.0%	0.39%	0.39%	0.0%	0.0%	0.0%

Table 4.18: The numbers of misclassification and rejection in different testing data groups, listed in line with each font and the entire data set, together with the corresponding recognition, rejection, and error rates of the latter for a set of multi-font Chinese characters of similar character subsets. The pairing is conducted after the input data set has been divided first into 8 subgroups.

	Testing Data Group								
	0°	5°	15°	30°	45°	60°	90°	180°	270°
Song	208.16	173.61	175.02	174.62	177.12	174.28	208.15	208.16	208.15
Kai	208.27	173.87	174.99	173.00	175.72	173.35	208.27	208.27	208.27
Boldface	200.74	171.26	171.52	171.41	171.65	171.88	200.74	200.74	200.74
Entire Set	205.73	172.91	173.84	173.01	174.83	173.17	205.72	205.73	205.72

Table 4.19: The average largest similarity value (LSV) obtained during the associative recollection processes with different testing data groups for the recognition of a set of multi-font Chinese characters of similar character subsets. The results are listed on the basis of each font and the entire data set. The average LSV's may vary from 0 to 256. And pairing is conducted after the input data set has been divided first into 8 subgroups.

Chapter 5

Conclusions

Artificial neural networks are biologically inspired. They are under development mainly in an attempt to mimic the function of the human brain, their biological model, in solving some intelligent problems, such as pattern recognition. Neural networks have a series of advantages, the important one of which to pattern recognition is its insensitiveness to variations in inputs. This ability to see through noise and distortion is the most desired characteristic of a pattern classifier, and explains why it has gained considerable interest in this area.

To date, the application of neural networks on character recognition has been studied widely, mainly for their use as pattern classifiers. Many successes are reported especially for the recognition of numerals and letters of the English alphabet. Also, the investigation of their application in that of Chinese characters has already started. The networks tried involve the single layer perceptron, back propagation model, neocognitron, DYSTAL network, and so on. However, the ability of some models which are successful in recognizing alphameric characters, like back propaga-

tion model, neocognitron *etc.*, is confined by the number of pattern categories they are dealing with currently. Also, the experiment of some other models with large amounts of classes is still not convincing, particularly due to the testing data's variability.

In this thesis, the usability of another neural network model as pattern classifier has been investigated. The model is associative memory network, and it is studied for the potential to solve the recognition problems involving a large number of categories, such as that of Chinese characters. Associative memory network comes into our mind for two reasons. One of them is the associative nature of human beings in information retrieval. Besides, technically, there is a similarity between the processes of associative mapping and pattern recognition.

When associative memory is used as a pattern classifier, the network will have several aspects different from other instances of its application, like the one for information recovery. Therefore, it is called associative memory classifier specifically. An AMC performs pattern classification based on information memorization and recollection. Its robustness against variations relies on the collective responding nature of the entire network. Hence, classifying a large number of categories means there are large amounts of memory items to be stored. Also, here the network is employed for classification purpose. As long as the recollected information is evident enough to make a right ruling, how serious it has been distorted really does not matter. Furthermore, unlike an information recovery network, the output vector in each associated pattern pair is not defined in advance, and therefore subject to selection according to the need.

The associative memory network in use for pattern classification is a feed forward system. The basic problem in the construction of such an AMC is the selection of appropriate output vectors to be associated with input patterns, which is called an inner coding process, provided that the latter have been prescribed. Like other neural networks, AMC is also a data-driven system after the determination of its configuration. It has been found that although the weights of an AMC is composed of both the input and output vectors, its information retrieval behaviour is ultimately determined by the characteristics of its input patterns. However, the proper selection of inner codes may lead to approaching its classification ability restricted thereof.

To describe the suitability of a set of input patterns to AMC so that the output vectors can be selected accordingly, two measurements have been set up and used repeatedly in our study. They are the average correlation coefficient of the set of input patterns and the probability distribution of the components of these pattern vectors. The discussion on the choice of inner codes is conducted with an example of a set of real-life multi-font Chinese characters. Based on the characteristics of this set of input data, Hadamard vectors which are the row vectors of Hadamard transformation matrix are selected, and a fairly good classification performance has been observed from the AMC constructed therefrom.

Different classification effects produced by different sets of inner codes attracted our interest in search for better inner coding schemes. Theoretically, the existence of an optimal scheme is guaranteed by the finitude in output vector's dimensionality. Nevertheless, trying to acquire such a scheme through enumeration is computationally

infeasible for it turns out to be a non-polynomial problem. Normally, problems as such are tackled on the basis of individual case, *i.e.* to find the solution for a given set of input vectors based on its specific feature. Unfortunately for real-life data, it is ususally impossible to express its features mathematically.

Inner coding can be accomplished practically in a two stage manner. The first is to select a set of vectors of some specific property, and the next to find a method for associating each of these outputs with an input pattern. This second stage is named as a pairing process. Therefore, a proposed solution to the aforementioned dilemma is to develop an optimal pairing scheme based on the traits of the selected inner codes and also certain assumptions of the input data, and then apply it to the practical data with some remedial measures for a better pairing scheme. Such a strategy in better inner coding scheme seeking is tested by the recognition of the set of multi-font Chinese characters with Hadamard vectors used as inner codes. The results of our simulation reveal improvements in the classification performance of the AMC's constructed from the pairing schemes thus obtained.

In spite of the importance of inner coding to AMC's classification behaviour, a more significant improvement is expected if a proper change can be made on its input patterns. This idea comes from the following facts. First of all, it is the distinctiveness of input patterns that limits an AMC's classification ability in the final analysis. Next, there is usually a feature extraction process right before pattern classification in a recognition system, where the form of patterns may change. Therefore, in addition to invariant feature detection, the suitability of the finally obtained feature vector

to AMC should be taken into account when the latter is used as a pattern classifier. Under such circumstances, data reduction is not a question any longer because any neural network system is good at dealing with large amounts of data due to its parallelism nature in computation. This effects a change in the basic tasks of the feature extraction process. In this thesis, feature extraction algorithms have been designed under these principles for printed character recognition, and such an adjustment in its fundamental objectives has been justified by the simulation outcomes.

A pattern recognition system mainly contains two functional parts, *i.e.* feature extraction and pattern classification. In this thesis, a novel neural network classifier has been developed based on the model of associative memory, and the corresponding feature extraction method has also been studied. This associative memory classifier was originally designed to solve the recognition problem involving a large number of categories. Our experiments have already been conducted on a scale of more than two hundred classes. However, since the categories of Chinese characters are counted by the thousands, and also in consideration of the fact that problems as such are usually handled by multi-level classifiers, our network will have to deal with patterns of greater similarity if it is used in such systems. For this reason, a set of multi-font Chinese characters comprised of many similar character subgroups have been introduced in computer simulations, and the results demonstrate the recognition capability of our system in dealing with highly similar character patterns in a large number of categories.

Appendix A

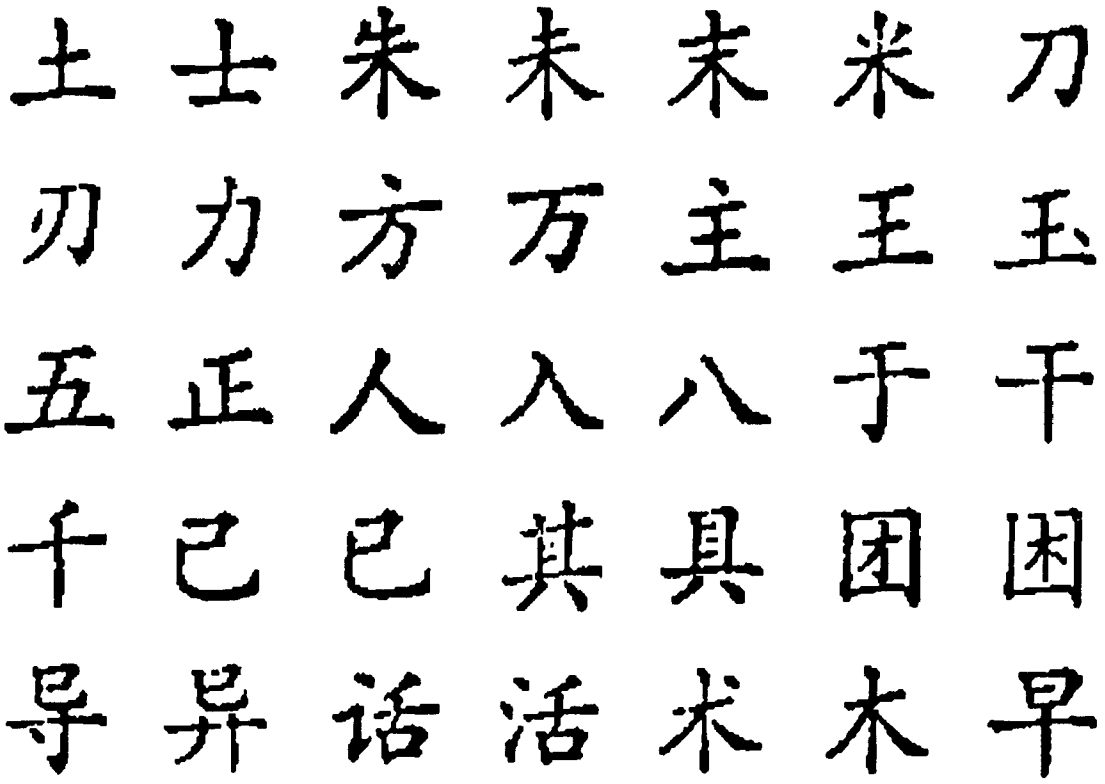
The Set of Common Chinese Characters in Use

的 一 是 在 了 不 和
有 大 这 主 中 人 上
为 们 地 个 用 工 时
要 动 国 产 以 我 到
他 会 作 来 分 生 对

年出说多机高定
就可面而小电家
义民行革加本实
级部同度社线得
下成进命自力党
学发方过后经长
于阶能种子也量
深

Appendix B

The Set of Similar Chinese Characters in Use



椎卢太衷句守粟
推阐大哀慨字粟
旦闸丈夫溉宁兔
且忙支矢概几免
夏证官失又儿厉
复乌宫火叉甸历
旱鸟户犬义旬字票

References

[1] V. K. Govindan and A. P. Shivaprasad, "Character recognition — A review," *Pattern Recognit.*, vol. 23, pp. 671–683, 1990.

[2] R. G. Casey and G. Nagy, "Automatic recognition of machine printed Chinese characters," *IEEE-TEC*, 1966.

[3] W. Stallings, "The morphology of Chinese characters: A survey of models and applications." *Computers and the Humanities*, vol. 9, pp. 13–24, 1975.

[4] W. Stallings, "Approaches to Chinese character recognition," *Pattern Recognit.*, vol. 8, pp. 87–98, 1976.

[5] C. Y. Suen, "Computer recognition of Kanji characters," *Proc. 1983 Conf. Text Processing with a Large Character Set*, pp. 429–435, 1983.

[6] L. D. Wu and J. W. Tai, "Some advances of pattern recognition in China," *Proc. 8th Int. Conf. Pattern Recognit.*, pp. 134–143, 1986.

[7] G. Nagy, "Chinese character recognition: A twenty-five year retrospective," *Proc. 9th Int. Conf. Pattern Recognit.*, vol. 1, pp. 163–167, 1988.

[8] C. Y. Suen, "Character recognition by computer and applications," in T. Z. Young and K. S. Fu (eds.), *Handbook of Pattern Recognition and Image Processing*, Academic Press, pp. 569-586, 1986.

[9] C. Y. Suen, "Processing of Chinese and oriental languages," *in press*, in A. Kent and J. G. Williams (eds.), *Encyclopedia of Computer Science and Technology*, Marcel Dekker Inc., New York, vol. 26, suppl. 11, pp. 333-347, 1992.

[10] L. Cheng, "Computer recognition of Chinese characters: An overview," *Proc. 1989 Int. Symp. Chinese Text Precessing*, vol. 1, pp. 2-8.

[11] W. C. P. Yu, H. H. Teh, H. B. Low, X. Yan, T. M. Ng and F. Gao, "A historical advancement of the Chinese language computing," *Computer Processing of Chinese & Oriental Languages*, vol. 4, pp. 57-81, 1988.

[12] Y. H. Pao, *Adaptive Pattern Recognition and Neural Networks*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1989.

[13] K. Nakata, Y. Nakano and Y. Uchikura, "Recognition of Chinese characters," *Proc. Conf. Machine Perception of Patterns of Pictures*, pp. 45-52, 1972.

[14] Y. X. Gu, Q. R. Wang and C. Y. Suen, " Application of multilayer decision tree in computer recognition of Chinese characters," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-5, pp. 83 89, 1983.

[15] P. P. Wang and R. C. Shiau, "Machine recognition of printed Chinese characters via transformation algorithms," *Pattern Recognit.*, vol. 5, pp. 303-321, 1973.

[16] K. Sakai, S. Hirai, T. Kawada, S. Amano and K. Mori, "An optical Chinese character reader," *Proc. 3nd Int. Joint Conf. Pattern Recognit.*, pp. 122-126, 1976.

[17] M. Yasuda and H. Fujisawa, "An improvement of correlation method for character recognition," *J. IECE Japan*, vol. J62-D, pp. 217-224, 1979.

[18] T. Saito, H. Yamada and K. Yamamoto, "An analysis of handprinted Chinese characters," *J. IECE Japan*, vol. J65-D, pp. 550-557, 1982.

[19] A. Shio and K. Komori, "A stroke extraction method for handprinted Chinese character recognition," *Trans. IECE Japan*, vol. IE80-14, pp. 83-90, 1980.

[20] W. H. Hsu, K. Takahashi and H. Fujita, "An expansion of pen movement stroke extraction method to muiltifont Chinese Character recognition," *Trans. IECE Japan*, vol. J65-D, pp. 1159-1166, 1982.

[21] Y. T. Wang and W. H. Hsu, "Stroke extraction and matching methods for handwritten Chinese characters recognition," *Proc. 1984 Int. Comput. Symp.*, pp. 441-446, 1984.

[22] M. Umeda, "A multifont printed Chinese character reader," *Trans. IECE Japan*, vol. J64-D, pp. 908-915, 1984.

[23] H. A. Glucksman, "Classification of mixed font alphabetics by characteristic loci," *Dig. 1st Annu. IEEE Comput. Conf.*, pp. 137-141, 1976.

[24] X. Zhang and C. D. Yan, "Feature point method in CCR and its application," *J. Chinese Information Processing*, vol. 1, pp. 13–19, 1987.

[25] F. H. Cheng, W. H. Hsu and M. Y. Chen, "Recognition of handwritten Chinese characters by modified Hough transform techniques," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI–11, pp. 429–439, 1989.

[26] Q. R. Wang and C. Y. Suen, "Analysis and design of a decision tree based on entropy reduction and its application to large character set recognition," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI–6, pp. 406–417, 1984.

[27] Q. R. Wang and C. Y. Suen, "Large tree classifier with heuristic search and global training," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI–9, pp. 91 102, 1987.

[28] S. Rasoul Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE Trans. Syst. Man Cyber.*, vol. SMC–21, pp. 660–674, 1991.

[29] R. P. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Magazine*, vol. 4, no. 2, pp. 4–22, 1987.

[30] R. P. Lippmann, "Pattern classification using neural networks," *IEEE Communications Magazine*, vol. 27, no. 11, pp. 47–64, 1989.

[31] T. F. Pawlicki, D. S. Lee, J. J. Hull and S. N. Srihari, "Neural network models and their application to handwritten digit recognition," *Proc. IEEE 2nd Int. Conf. Neural Netw.*, vol. 2, pp. 63–70, 1988.

[32] D. L. Bisset, E. Filho and M. C. Fairhurst, "A comparative study of neural network structures for practical application in a pattern recognition environment," *Proc. IEE 1st Int. Conf. Artificial Neural Netw.*, pp. 378-382, 1989.

[33] R. E. Howard, B. Boser, J. S. Denker, H. P. Graf, D. Henderson, W. Hubbard, L. D. Jachel, Y. Le Cun and H. S. Baird, "Optical character recognition: A technology driver for neural networks," *Proc. IEEE Symp. Circuits Syst.*, vol. 3, pp. 2433-2436, 1990.

[34] A. Krzyzak, W. Dai and C. Y. Suen, "Unconstrained handwritten character classification using modified backpropagation model," in C. Y. Suen (ed.), *Frontiers in Handwriting Recognition*, CENPARMI, Montreal, pp. 155-165, 1990.

[35] A. Krzyzak, W. Dai and C. Y. Suen, "Classification of large set of handwritten characters using modified back propagation model," *Proc. Int. Joint Conf. Neural Netw., San Diego, CA, USA*, vol. 3, pp. 225-232, 1990.

[36] W. Dai, A. Krzyzak and C. Y. Suen, "On the recognition of handwritten characters using neural network," *Proc. Vision Interface '90*, pp. 84-93, 1990.

[37] Y. Le Cun, "Generalization and network design strategies," in R. Pfeifer, Z. Schreter, F. Fogelman-Soulié and L. Steels (eds.), *Connectionism in Perspective*, Elsevier, Zurich, Switzerland, pp. 143-155, 1989.

[38] Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard and L. D. Jackel, "Backpropagation applied to Handwritten zip code recognition," *Neural Comput.*, vol. 1, pp. 541-551, 1989.

[39] Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel and H. S. Baird, "Constrained neural network for unconstrained handwritten digit recognition," in C. Y. Suen (ed.), *Frontiers in Handwriting Recognition*, CENPARMI, Montreal, pp. 145–152, 1990.

[40] Y. Le Cun, O. Matan, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel and H. S. Baird, "Handwritten zip code recognition with multilayer networks," *Proc. 10th Int. Conf. Pattern Recognit.*, vol. 2, pp. 35-40, 1990.

[41] Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in D. S. Touretzky (ed.), *Advances in Neural Information Processing Systems 2*. Morgan Kaufmann, San Mateo, CA, pp. 396–404, 1990.

[42] Y. Le Cun, L. D. Jackel, H. P. Graf, B. Boser, J. S. Denker, I. Guyon, D. Henderson, R. E. Howard, W. Hubbard and S. A. Solla, "Optical character recognition and neural-net chips," *Int. Neural Netw. Conf. (Paris)*, vol. 2, pp. 651 655, 1990.

[43] L. D. Jackel, B. Boser, J. S. Denker, H. P. Graf, Y. Le Cun, I. Guyon, D. Henderson, R. E. Howard, W. Hubbard and S. A. Solla, "Hardware requirements for neural-net optical character recognition," *Proc. Int. Joint Conf. Neural Netw., San Diego, CA, USA*, vol. 2, pp. 855-861, 1990.

[44] L. D. Jackel, B. Boser, J. S. Denker, H. P. Graf, Y. Le Cun, I. Guyon, D. Henderson, R. E. Howard, W. Hubbard, O. Matan and S. A. Solla, "Hardware for neural-net optical character recognition," in R. Eckmiller (ed.), *Advanced Neural Computers*, Elsevier Science Publishers B.V. (North-Holland), pp. 193 200, 1990.

[45] K. Yamada, H. Kami, J. Tsukumo and T. Temma, "Handwritten numeral recognition by multi-layered neural wetwork with improved learning algorithm," *Proc. Int. Joint Conf. Neural Netw., Washington, DC, USA*, vol. 2, pp. 259 266, 1989.

[46] D. J. Burr, "A neural network digit recognizer," *Proc. IEEE Int. Conf. Syst. Man. Cyber.*, vol. 1, pp. 1621-1625, 1986.

[47] D. J. Burr, "Experiments on neural net recognition of spoken and written text," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP 36, pp. 1162 1168, 1988.

[48] W. E. Weideman, M. T. Manry and H. C. Yau, "A comparison of a nearest neighbor classifier and a neural network for numeric handprint character recognition," *Proc. Int. Joint Conf. Neural Netw., Washington, DC, USA*, vol. 1, pp. 117-120, 1989.

[49] U. Kreßel, J. Franke and J. Schürmann, "Polynomial classifier versus multilayer perceptron," *Technical Report of Research Center Ulm, Daimler-Benz AG*, 1990.

[50] E. Gullichsen and E. Chang, "Pattern Classification by neural network: An experimental system for icon recognition," *Proc. IEEE 1st Int. Conf. Neural Netw.*, vol. 4, pp. 725-732, 1987.

[51] A, Khotanzad and J. H. Lu, "Distortion invariant character recognition by a multi-layer perceptron and back-propagation learning," *Proc. IEEE 2nd Int. Conf. Neural Netw.*, vol. 1, pp. 625-632, 1988.

[52] A. Rajavelu, M. T. Musavi and M. V. Shirvaikar, "A neural network approach to character recognition," *Neural Netw.*, vol. 2, pp. 387-393, 1989.

[53] M. Hosokawa, S. Omatu and M. Fukumi, "A new approach for pattern recognition by neural networks with scramblers," *Proc. Int. Joint Conf. Neural Netw., Washington, DC, USA*, vol. 1, pp. 183-188, 1989.

[54] M. Oita, M. Takahashi, S. Tai, K. Kojima and K. Kyuma, "Character recognition using a dynamic opto-electronic neural network with unipolar binary weights," *Proc. Int. Joint Conf. Neural Netw., San Diego, CA, USA*, vol. 2, pp. 789-794, 1990.

[55] S. B. Cho and J. H. Kim, "Hierarchically structured neural networks for printed Hangul character recognition," *Proc. Int. Joint Conf. Neural Netw., San Diego, CA, USA*, vol. 1, pp. 265-270, 1990.

[56] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning internal representations by error propagation," in D. E. Rumelhart, J. L. McClelland and PDP Research Group (eds.), *Parallel Distributed Processing: Explorations in*

the Microstructure of Cognition, Vol 1: Foundations. MIT Press, Cambridge, pp. 318–362, 1986.

[57] B. Boser, E. Sächinger, J. Bromley, Y. Le Cun, R. E. Howard and L. D. Jackel, "An analog neural network processor and its application to high-speed character recognition," *Proc. Int. Joint Conf. Neural Netw., Seattle, WA, USA*, vol. 1, pp. 415–420, 1991.

[58] G. Mirchandani and W. Cao, "On hidden nodes for neural nets," *IEEE Trans. Circuits Syst.*, vol. CAS-36, pp. 661-665, 1989.

[59] Y. Mori and K. Yokosawa, "Neural networks that learn to discriminate similar Kanji characters," in D. S. Touretzky (ed.), *Advances in Neural Information Processing Systems 1*, Morgan Kaufmann, San Mateo, CA, pp. 332 339, 1989.

[60] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biol. Cybern.*, vol. 36, pp. 193-202, 1980.

[61] K. Fukushima, "Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position," *Pattern Recognit.*, vol. 15, pp. 455 469, 1982.

[62] K. Fukushima, "Neocognitron: A hierarchical neural network capable of visual pattern recognition," *Neural Netw.*, vol. 1, pp. 119 130, 1988.

[63] K. Fukushima, "Analysis of the process of pattern recognition by the neocognitron," *Neural Netw.*, vol. 2, pp. 413-420, 1989.

[61] K. Fukushima and N. Wake, "Alphanumeric character recognition by the neocognitron," in R. Eckmiller (ed.), *Advanced Neural Computers*, Elsevier Science Publishers B.V. (North-Holland), pp. 263–270, 1990.

[65] K. Fukushima and N. Wake, "Handwritten alphanumeric character recognition by the neocognitron," *IEEE Trans. Neural Netw.*, vol. 2, pp. 355-365, 1991.

[66] T. Ito, K. Fukushima and S. Miyake, "Realization of a neural network model neocognitron on a hypercube parallel computer," *Int. J. High Speed Comput.*, vol. 2, pp. 1-16, 1990.

[67] T. Ito, K. Fukushima and S. Miyake, "Examination of implementing a neural network on a parallel computer — neocognitron on NCUBE," *Syst. Comput. Jpn. (USA)*, vol. 22, no. 6, pp. 1-9, 1991.

[68] K. Fukushima and T. Imagawa, "Recognition and segmentation of connected characters in cursive handwriting with selective attention," in T. Kohonen, K. Mäkisara, O. Simula and J. Kangas (eds.), *Artificial Neural Networks*, Elsevier Science Publishers B.V. (North-Holland), vol. 1, pp. 105–110, 1991.

[69] K. Fukushima, T. Imagawa, and E. Ashida, "Character recognition with selective attention," *Proc. Int. Joint Conf. Neural Netw., Seattle, WA, USA*, vol. 1, pp. 593-598, 1991.

[70] C. L. Wilson, R. A. Wilkinson and M. D. Garris, "Self-organizing neural network character recognition on a massively parallel computer," *Proc. Int. Joint Conf. Neural Netw., San Diego, CA, USA*, vol. 2, pp. 325–329, 1990.

[71] M. D. Garris, R. A. Wilkinson and C. L. Wilson, "Methods for enhancing neural network handwritten character recognition," *Proc. Int. Joint Conf. Neural Netw., Seattle, WA, USA*, vol. 1, pp. 695–700, 1991.

[72] T. Shimada, K. Nishimura and K. Haruki, "A new self-organizing method and its application to handwritten digit recognition," *Proc. Int. Joint Conf. Neural Netw., Seattle, WA, USA*, vol. 1, pp. 275–281, 1991.

[73] Y. Hirai and Y. Tsukui, "Position independent neuro pattern matching and its application to handwritten numerical character recognition," *Proc. Int. Joint Conf. Neural Netw., San Diego, CA, USA*, vol. 3, pp. 695 701, 1990.

[74] D. L. Alkon, K. T. Blackwell, G. S. Barbour, A. K. Rigler and T. P. Vogl, "Pattern recognition by an artificial network derived from biologic neuronal systems," *Biol. Cybern.*, vol. 62, pp. 363–376, 1990.

[75] D. L. Alkon, T. P. Vogl and K. T. Blackwell, "Artificial learning networks derived from biologic neural systems," in P. Antognetti and V. Milutinović eds., *Neural Networks: Concepts, Applications, and Implementations*, vol. 4, pp. 24–46, 1991.

[76] T. P. Vogl, K. L. Blackwell, S. D. Hyman, G. S. Barbour and D. L. Alkon, "Classification of hand-written digits and Japanese Kanji," *Proc. Int. Joint Conf. Neural Netw., Seattle, WA, USA*, vol. 1, pp. 97–102, 1991.

[77] K. T. Blackwell, T. P. Vogl, S. D. Hyman, G. S. Barbour and D. L. Alkon, "A new approach to hand-written character recognition," *Pattern Recognit.*, Vol. 25, pp. 655 666, 1992.

[78] W. Poechmueller and M. Glesner, "Handwritten pattern recognition with a binary associative memory," *Proc. Int. Joint Conf. Neural Netw., San Diego, CA, USA*, vol. 1, pp. 873–878, 1990.

[79] J. Homma, Y. Kosugi and T. Sato, "Evaluation and improvement of associative memory for pattern recognition," *Applied Optics*, vol. 29, pp. 1675–1681, 1990.

[80] L. O. Chua and L. Yang, "Cellular neural networks: Applications," *IEEE Trans. Circuits Syst.*, vol. CAS-35, pp. 1273-1290, 1988.

[81] J. H. Li, A. N. Michel and W. Porod, "Analysis and synthesis of a class of neural networks: Linear systems operating on a closed hypercube," *IEEE Trans. Circuits Syst.*, vol. CAS-36, pp. 1405-1422, 1989.

[82] Y. Mori and K. Joe, "A large-scale neural network which recognizes handwritten Kanji characters," in D. S. Touretzky (ed.), *Advances in Neural Information Processing Systems 2*, Morgan Kaufmann, San Mateo, CA, pp. 415–422, 1990.

[83] Y. Kimura, "Distorted handwritten Kanji character pattern recognition by a learning algorithm mininizing output variation," *Proc. Int. Joint Conf. Neural Netw., Seattle, WA, USA*, vol. 1, pp. 103–106, 1991.

[84] H. T. Yen, K. Y. Huang and C. H. Chen, "Recognition of printed Chinese characters using multi-layer perceptron and Walsh functions," *Proc. Int. Conf. Computer Processing of Chinese & Oriental Languages*, pp. 98 103. 1991.

[85] J. F. Wang, H. D. Chang and J. H. Tzeng, "Handwritten Chinese radical recognition via neural networks," *Proc. Int. Conf. Computer Processing of Chinese & Oriental Languages*, pp. 92-97, 1991.

[86] L. Y. Tseng and T. H. Huang, "Recognition of hand-printed Chinese characters based on backpropagation neural network," *Proc. Int. Conf. Computer Processing of Chinese & Oriental Languages*, pp. 86-91, 1991.

[87] B. S. Jeng, S. W. Sun, C. J. Lee, K. H. Shyu, F. H. Liu and T. M. Wu, "Clustering and classification for Chinese character recognition," *Proc. SPIE Vol. 1199: Visual Communications and Image Processing IV*, pp. 1324 1331, 1989.

[88] B. S. Jeng, S. W. Sun, C. J. Lee, T. M. Wu and M. W. Chang, "Chinese character recognition with neural nets classifier," *Proc. Int. Conf. Acoust., Speech, Signal Processing*, vol. 4, pp. 2125–2128, 1990.

[89] B. S. Jeng, S. W. Sun, K. S. Chou, T. M. Wu, C. J. Lee and C. H. Shih, "A further study on clustering techniques for Chinese character recognition," *Proc. Int. Conf. Computer Processing of Chinese & Oriental Languages*, pp. 158 162, 1991.

[90] B. Widrow and M. A. Lehr, "30 years of adaptive neural networks: Perceptron, Madaline, and backpropagation," *Proc. IEEE*, vol. 78, pp. 1415–1442, 1990.

[91] S. D. Wang and C. C. Pan, "A neural network approach for Chinese character recognition," *Proc. Int. Joint Conf. Neural Netw., San Diego, CA, USA*, vol. 2, pp. 917-923, 1990.

[92] Y. Yao, "A neural network model of CAAM and its application to handprinted Chinese character recognition," *Proc. IEEE 1st Int. Conf. Neural Netw.*, vol. 3, pp. 309 316, 1987.

[93] Y. Yao, "Handprinted Chinese character recognition via neural networks," *Pattern Recognit. Lett.*, vol. 7, pp. 19–25, 1988.

[94] S. D. Hyman, T. P. Vogl, K. T. Blackwell, G. S. Barbour, J. M. Irvine and D. L. Alkon, "Classification of Japanese Kanji using principal component analysis as a preprocessor to an artificial neural network," *Proc. Int. Joint Conf. Neural Netw., Seattle, WA, USA*, vol. 1, pp. 233–238, 1991.

[95] A. Iwata, T. Tohma, H. Matsuo and N. Suzumura, "A large scale neural network 'CombNET' and its application to Chinese character recognition," *Proc. Int. Neural Netw. Conf., Paris*, vol. 1, pp. 83–86, 1990.

[96] H.P. Graf, W. Hubbard, L. D. Jackel and P. G. N. de Vegvar, "A CMOS associative memory chip," *Proc. IEEE 1st Int. Conf. Neural Netw.*, vol. 3, pp. 461 468, 1987.

[97] L. D. Jackel, H. P. Graf, W. Hubbard, J. S. Denker and D. Henderson, "An application of neural net chips: Handwritten digit recognition," *Proc. IEEE 2nd Int. Conf. Neural Netw.*, vol. 2, pp. 107-115, 1988.

[98] J. S. Denker, W. R. Gardner, H. P. Graf, D. Henderson, R. E. Howard, W.Hubbard, L. D. Jackel, H. S. Baird and I. Guyon, "Neural network recognizer for hand-written zip code digits," in D. S. Touretzky (ed.), *Advances in Neural Information Processing Systems 1*, Morgan Kaufmann, San Mateo, CA, pp. 323–331, 1989.

[99] Y. Le Cun, L. D. Jackel, B. Boser, J. S. Denker, H. P. Graf, I. Guyon, D. Henderson, R. E. Howard and W. Hubbard, "Handwritten digit recognition: Applications of neural netword chips and automatic learning," *IEEE Communications Magazine*, vol. 27, no. 11, pp. 41-46, 1989.

[100] X. Y. Zhu, Y. Iwase, T. Jimbo and M. Umeno, "A model for pattern recognition," *The Transactions of the Institute of Electronics, Information and Communication Engineers*, vol. E72, pp. 888-890, 1989.

[101] X. Y. Zhu, K. Yamauchi, T. Jimbo and M. Umeno, "Handwritten character recognition by a layered neural network," *Syst. Comput. Jpn. (USA)*, vol. 21, no. 13, pp. 88-97, 1990.

[102] J. Rubner and K. Schulten, "Development of feature detectors by self organization," *Biol. Cybern.*, vol. 62, pp. 193-199, 1990.

[103] M. Zhang, C. Y. Suen and T. D. Bui, "Multi-font Chinese character recognition with associative memory network," in T. Kohonen, K. Mäkisara, O. Simula and J. Kangas (eds.), *Artificial Neural Networks*, Elsevier Science Publishers B.V. (North-Holland), vol. 2, pp. 1199-1202, 1991.

[104] M. Zhang, C. Y. Suen and T. D. Bui, "Associative memory in pattern recognition," *Proc. Canadian Conf. Elec. Computer Eng.*, vol. 1, pp. 31.1.1-31.1.4, 1991.

[105] M. Zhang, Y. Y. Tang, C. Y. Suen and T. D. Bui, "A study on associating schemes in associative memory classifier," *Proc. IEEE Int. Symp. Information Theory*, p. 372, 1991.

[106] M. Zhang, C. Y. Suen and T. D. Bui, "A pairing strategy in associative memory classifier," *Proc. Int. Joint Conf. Neural Netw., Seattle, WA, USA*, vol. 2, p. A-901, 1991.

[107] M. Zhang, C. Y. Suen and T. D. Bui, "An optimal pairing scheme in associative memory classifier and its application in character recognition," *Proc. 11th Int. Conf. Pattern Recognit.*, vol. 2, pp. 50-53, 1992.

[108] M. Zhang, C. Y. Suen and T. D. Bui, "A distortion invariant feature extraction algorithm used with associative memory classifier," *Proc. Int. Joint Conf. Neural Netw., Baltimore, MD, USA*, 1992.

[109] M. Zhang, C. Y. Suen and T. D. Bui, "Feature extraction for accommodating pattern vectors to a neural classifier," *Proc. Canadian Conf. Elec. Computer Eng.*, vol. 2, pp. MM10.7.1-MM10.7.4, 1992.

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

[111] T. Kohonen, *Self-Organization and Associative Memory*. Springer-Verlag, Berlin, 1989.

[112] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. Natl. Acad. Sci. USA*, vol. 79, pp. 2554-2558, 1982.

[113] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proc. Natl. Acad. Sci. USA*, vol. 81, pp. 3088-3092, 1981.

[114] B. Kosko, "Adaptive bidirectional associative memories," *Appl. Optics*, vol. 26, pp. 4947-4960, 1987.

[115] B. Kosko, "Bidirectional Associative Memories," *IEEE Trans. Syst. Man Cyber.*, vol. SMC 18, pp. 49-60, 1988.

[116] R. C. Gonzalez and P. Wintz, *Digital Image Processing*. Addison-wesley Publishing Company, Reading, Massachusetts, 1987.

[117] J. T. Tou and R. C. Gonzalez, *Pattern Recognition Principles.* Addison-Wesley Publishing Company, Reading, Massachusetts, 1974.