Combination of Multiple Classifiers for the Recognition of Totally Unconstrained Handwritten Numerals

Yea-Shuan Huang

A Thesis

in

The Department

of

Computer Science

Presented in Partial Fulfillment of the Requirements

for the Degree of Doctor of Philosophy at Concordia University

Montréal, Québec, Canada

ISBN 0-612-01287-5

ABSTRACT

Combination of Multiple Classifiers for the Recognition of Totally Unconstrained Handwritten Numerals

Yea-Shuan Huang, Ph.D.

Concordia University, 1991

Due to different writing styles and various kinds of noise, the recognition of handwritten numerals is an extremely challenging problem. Recently, a new approach has emerged to tackle this problem by the use of multiple classifiers. This method is called "Combination of Multiple Experts" (CME). It combines individual classification decisions to derive the final decisions. This thesis focusses on methodologies which lead to efficient and effective decision combination schemes.

In general, the output information supplied by various classifiers can be divided into two levels: (1) The abstract level: a classifier only outputs a unique class; and (2) The measurement level: a classifier assigns a measurement value to each class to indicate how closely a certain class corresponds to the input pattern. Because of the different nature of the two levels of output information (one is discrete and finite, and the other one is continuous and infinite), intrinsically there are two kinds of combination approaches: *abstract-level CME* and *measurement-level CME*.

For abstract-level CME, a novel combination model is proposed, *i.e.* the Behavior-Knowledge Space (BKS) method. Many advantageous properties have been derived

from this method: most importantly, in theory the BKS method is able to produce the highest recognition accuracy for the combination of abstract-level classifiers. For measurement-level CME, individual classifiers can be regarded as feature extractors. A neural network approach based on a multi-layer perceptron is proposed to perform the combination function, because the multi-layer perceptron has been shown to be effective in solving various pattern recognition problems. Strategies on improving multi-layer perceptrons are presented.

Since the basic components of a multi-classifier system are the classifiers, it is essential to construct classifiers with high recognition accuracy. In this research, two directions are taken to pursue this goal. The first is to apply the concept of CME to design new classifiers by using subsets of a large set of features. The second is to improve the recognition accuracy of the commonly-used nearest neighbor classifier by finding better representative prototypes. Obviously, the new classifier design techniques and the improvement of existing classifiers will further strengthen CME.

The efficiency of the proposed methods has been demonstrated in a series of experiments with a large data base of handwritten numerals. The results indicate that a multi-classifier recognition system outperforms the individual classifiers, and is able to achieve a very high recognition performance. It appears, therefore, that CME is a promising avenue to develop human-compatible and highly reliable OCR machines.

To Hsiu-Wen Fan, my dear wife, for her love and support

To Alice Huang, my lovely daughter, for the best gift from God

To my parents, for their care and encouragement

ACKNOWLEDGEMENTS

I would like to thank Professor Ching Y. Suen, founder and Director of the Centre of Pattern Recognition and Machine Intelligence (CENPARMI). Professor Suen is my supervisor. He does not only provide me a comfortable and convenient research environment, but he also enthusiastically directed me to pursue this research. It is his supervision and support that has made this thesis possible.

I am also grateful to Mr. Jürgen Franke, senior researcher of Daimler-Benz Research Center. During his visit to CENPARMI, he generously shared with me his professional experience in the course of this research.

Special thanks to Mr. Lo T. Tu, leader of Chinese Character Recognition project at Industrial Technology Research Institute, Taiwan. He kindly provided classification decisions of several classifiers so that a series of experiments could be performed, which are essential to the verification of the efficiency of our proposed methods.

I am also indebted to Dr. Qiang Gan, Chairman of the Biomedical Engineering Department at Southeast University, China. Many ideas have been triggered out of discussions with him, and through him, I gained a deeper understanding of neural networks.

Special thanks to Dr. Ke Liu, professor of the Computer Science Department at Nanjing University of Science and Technology, China. His enthusiastic involvement along the course of this research strengthened my motivation to attain my best.

Without his cooperation, this thesis will not achieve such a complete scale.

I would like to express my sincere gratitude to Dr. Tony Kasvand, Dr. Sabine Bergler, Dr. Tao Li, and Dr. Louisa Lam. They gave me very useful directions to pursue this research, and their challenging questions made the theoretical part of this thesis more complete.

I also thank Mr. Willian Wong and the analysts of the Department of Computer Science for providing a stable and convenient environment so that our experiments could be efficiently performed.

Special thanks to Mr. Al Bloch for his consistent help to improve my English in writing and speaking. Also, his devotion to proofread this thesis.

Thanks to all the members of CENPARMI who contributed to a friendly atmosphere and open research environment.

My very thanks go to Industrial Technology and Research Institute (ITRI), Taiwan. Without the financial support from ITRI, I would be unable to pursue my Ph.D. study and complete this thesis.

This research has been supported by research grants from the Natural Sciences and Engineering Research Council of Canada, the National Networks of Centres of Excellence Research Program of Canada, the Ministry of Education of Quebec, Bell Quebec, and the Chinese Character Recognition Project (project no. 35N1100) supported by MOEA, Taiwan, R.O.C.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Handwriting recognition by computer has been a subject of intense research for many years. This is driven by the strong desire of the researchers to take up the challenge of developing algorithms comparable to human performance, and by the numerous possible applications in data processing. To date, many character recognition systems have been developed, but more work is still required before human performance can be matched in a meaningful way. Recently, a promising direction was suggested that instead of a single classifier, a number of classifiers can be used in parallel to tackle the recognition problem. This has the advantage that different features and classification procedures can be used simultaneously to complement one another resulting in an enhancement of their strengths and a reduction of their weaknesses. Two key tasks are involved in this approach, (1) to choose the appropriate features or classification

procedures which can be used in a multiple-classifier system, and (2) to design the method of combining decisions which can effectively take advantage of the strengths of individual classifiers and avoid their weaknesses. This thesis focuses mainly on research in the second task.

Broadly speaking, this thesis contains three parts. The first part begins with an introduction of a conventional character recognition system, followed by a brief review of both character features and classification procedures, and a description of the motivations of a new recognition trend with multiple classifiers. The basic terminology related to this research and the formal specification of the combination of multiple classifiers are also described.

The second part begins by introducing the previous research work on combination procedures, their advantages and disadvantages, and then our research effort in proposing several new combination procedures. Interestingly, with a different viewpoint on the roles of individual classifiers, the combination problem turns out to be a generic pattern recognition problem. This realization enables us to utilize the existing pattern classification techniques to explore various techniques of combining multiple classifiers. Finally, with three classifiers and a large numeral data base, a thorough comparison of different combination procedures is performed and described.

Since a combination problem is intrinsically a pattern recognition problem, the techniques of combining multiple classifiers in fact can be treated as new pattern

classification techniques. With this in mind, the last part of this thesis first describes our research in applying combination techniques to pattern classification. Two classifier design techniques are introduced: *recognition by parts* and *recognition by pair classifiers*. With an another consideration that because the basic component of CME is a classifier, and CME turns out to be a generic pattern classification problem, it is essentially important to improve the performance of the existing classification functions. In this part, we also describe our effort on the research to derive optimal prototypes for the nearest neighbor classification.

1.1 Basic Operations in Character Recognition Systems

Figure 1 shows the block diagram of a typical optical character recognition (OCR) system. Basically it performs three functions: *preprocessing*, *feature extraction*, and *pattern classification*.

1.1.1 Preprocessing

Preprocessing plays an important role in a pattern recognition system. In general, it consists of *binarization*, *segmentation*, *noise removal*, and *normalization* or *skeletonization*. An input character is scanned and digitized by an optical scanner to produce a gray-level digital image. Through binarization, the image is converted into a binary-level one (each pixel is either black or white) by using a threshold optimal to the processed image. The segmentation process specifies which areas of an image

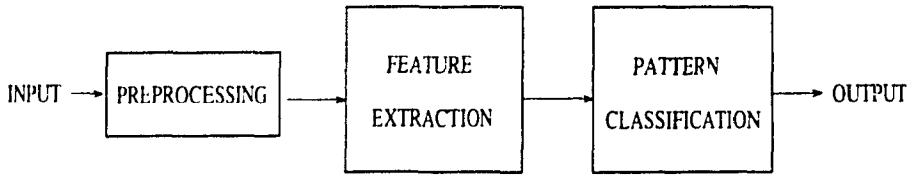


Figure 1: A conventional pattern recognition system.

contain isolated characters for recognition. The purpose of noise removal is to produce a better quality image by eliminating the tiny parts which do not belong to the image of a character or connecting the broken parts of character strokes. Normalization or skeletonization transforms each pattern into an image with a fixed size or a single-pixel stroke-width of image, which will enhance both feature extraction and pattern classification.

1.1.2 Feature Extraction

Till now, it is still not clear how humans are able to recognize characters effortlessly even when they are written carelessly and sloppily. But it has been observed in research [1, 2] that when humans look at an unfamiliar scene or image, or when they are reading characters, their eyes tend to seek out "high information content" points, and their visual attention shifts from feature to feature as they acquaint themselves with the pattern. Then a direct problem emerges: "What are features?". According to Nadler and Eric [3], "Features are functions of the measurements performed on a class of objects which enable that class to be distinguished from other classes". Since

features are important to character recognition and no one really knows exactly which features humans use to recognize characters, feature extraction has always been the most attractive and challenging OCR research topic. Researchers continue to search for more efficient and effective features for the recognition of characters. In general, good character features are characterized by small feature variances within the same character class, and large feature variances among different character classes. In other words, effective features should be both stable and distinctive.

Many character features have been developed. Broadly speaking, they are derived from two main feature detection schemes [4]:

1. Global analysis, and

2. Structural analysis.

A. Global Analysis

In general, there are three methodologies commonly used in global analysis, namely *distribution of pixels*, *transformation*, and *physical measurements*. Usually, features extracted from such analysis are represented by feature vectors.

1) Distribution of pixels

Conventionally, a character image is represented by a two-dimensional binary matrix where *pixel* is the basic element. With specific but non-random arrangements of pixels, characters are deliberately constructed in different shapes. Heuristically, various measurements of pixel distributions could be used as features for classification.

Usually, these measurements are size-dependent; therefore they must be normalized into fixed dimensions before matching them against the reference patterns. Many pixel-distribution features have been developed. For example, Shimura [5] normalized a character image into a mesh with binary elements. Using Hamming distance, it is easy to measure the similarity between an input mesh and the references. Instead of using all elements in a mesh, Bledsoe and Browing [6] chose a subset of them for comparison. Kwon *et al.* [7] described a feature which counts the number of times that a line crosses from a white pixel to a black pixel of the entire character image along the direction of the line. The crossing feature is a special aspect of *characteristic loci* originally designed by Glucksman [8] for the recognition of multi-font printed alphabets. Projections of pixel distribution with respect to horizontal and vertical directions offer another type of features.

2) Transformation

It is well known that frequency information is very important in one-dimensional signal processing. Many transformation functions have been developed to extract frequency related features, such as Fourier transform [9] or Hadamard transform [10]. Since a character image is a two-dimensional signal, it is believed that frequency information is also important for character classification. Besides frequency features, spatial and spectrum features can also be extracted by various transformation techniques such as Walsh transform [11, 12] or Kahunen-Loeve transform [13]. As a

matter of fact, a transformation operation can be performed not only on the original image but also on different measurement data. In general, various transformations are useful because of two reasons: a) features can be expressed in forms other than the original measurements, as Fourier transform can convert the spatial image into a series of coefficient expansions in the frequency domain; b) features can be expressed by other orthogonal axes which have more discrimination power for pattern classification, *e.g.* K-L transform can rotate the axes of feature space to become orthogonal to each other, constituting a "decorrelation". Another advantage of transformation is that the feature dimension can be compressed significantly, so that only stable but distinctive features remain.

3) Physical Measurements

The width and height of a character image are also important features. Related to them, information such as aspect ratio and area covered by black and white elements can be computed. Such measurements are useful in distinguishing pairs of ambiguous characters. For example, Duda [14] and Suen [15] used some physical measurements to distinguish confusing pairs of characters $\{8, B\}$, and $\{U, V\}$, respectively.

B. Structural Analysis

Three methodologies have been used in structural analysis, namely *line segments and edges, outline of character*, and *center-line of character*. All three methodologies produce a line representation of the character.

1) Edge and Line Segments

A character can be regarded as a picture or a graph. When a character is regarded as a picture, it is important to extract the edge information because edges contain the richest information of the picture. This can be justified by recalling that in most image processing textbooks, example pictures are easily perceptible by only showing their edge pixels. On the other hand, when a character is regarded as a graph, it is convenient to use line segments to represent the graph. In general, edges are detected by moving a mask window across the entire image. Many masks have been designed for edge detection such as Robert operator [16], Sobel operator [17] and Laplacian operator [9]. The result of such detection methods is a list of edges. In order to describe a graph, a further step of linking or grouping the edges into piecewise line segments is necessary. Once the edges and line segments have been constructed, other geometrical and topological features can be deduced as well, such as concavities, loops and T-joints.

2) Contour

It is well known that the contour carries valuable information about two-dimensional image patterns. Therefore, contour features are often used in character recognition. Contour can be extracted by tracing the character boundary in a clockwise or counter-clockwise direction. After the contour of the character has been found, information

on length, orientation and position of all contour points is obtained. Many other features can also be deduced from the contour, such as end points, lengths or inclination of line segments, and concavities and convexities.

3) Centre Line

It is believed that the width of character strokes is not critical to human recognization of characters. Seemingly, the different stroke widths of characters are mainly a design consideration, the purpose of which is to display characters aesthetically to human perception. Therefore, for recognition, a character may be represented by a single-pixel stroke-width image. In general, we refer to the single-pixel stroke-width image of a character as the centre line of this character. By using the centre line, a character can be described by a line-segment graph, which again can be theoretically recognized by the parsing procedure in formal language. Therefore, centre line extraction also attracts much attention in OCR research. Terminologically, the process to produce the centre line of a pattern is called "thinning" or "skeletonization" [18]. After thinning, a character is expressed by the centre line (or skeleton) of the image with a single-pixel stroke width. A derived skeleton may be encoded by the famous chain-code algorithm, from which a large number of descriptive features can be generated, *e.g.*, the area enclosed in a closed chain code, first and second moments about the horizontal and vertical axes, and the distance between two points on the skeleton.

1.1.3 Pattern Classification

For a recognition system, the ultimate objective is to achieve a high recognition accuracy. Therefore, besides searching for distinctive features, research into efficient and effective classification techniques is essential. Terminologically, various kinds of classification methodologies can also be grouped into two categories:

1. Statistics-based classification, and

2. Syntactics-based classification.

In general, the classification models of the two categories are often designed by human programmers with heuristics concerning either the shapes of the distributions of character features or possible character variations. This constrains both the capability and flexibility of applying one classification function to different data sets. Recently, neural networks have resurfaced and attracted great attention in many research fields, such as biology, psychology, pattern recognition, artificial intelligence, and control theory. Due to their network-like structure, parallel computation, and automatic learning ability, neural networks seemingly can be discussed as the third category of classification methodologies.

For any specific application problem, various classification procedures could be developed; usually each one has a different degree of success, but quite probably none of them is totally perfect, or even not as good as expected for practical applications.

A. Statistics-based classification

Statistical pattern recognition is a well-defined discipline, solidly grounded in mathematical statistics, with an extensive literature. In this approach, features describing a pattern are treated only as variables without considering their "meaning", and they are expressed in an ordered array called a "feature vector". In brief, statistics-based classification is metric and quantitative. Such an arrangement enables statistics-based classification to support computational analysis of interesting and useful characteristics such as error rates, convergence of learning algorithms, *etc.* The most commonly-used statistics-based classification techniques are *nearest neighbor*, *Bayesian*, *Perceptron* and *polynomial discrimination rules*.

It is well known in pattern recognition that without the complete knowledge of the class probability densities for the entire pattern or feature space, the nearest neighbor rule [19] is generally regarded as the best classification rule, with an asymptotic error rate less than twice the Bayes rate [19], the smallest possible incorrect classification rate in the current feature space. According to the nearest neighbor rule, a pattern is assigned to the class of the nearest prototype selected from training samples. Although the nearest neighbor rule is simple and powerful, its implementation is computationally expensive in terms of storage space and computation time, if the number of prototypes is large. Many attempts have been made in the past to

alleviate the computational burden of nearest neighbor classifiers. In general, traditional research on nearest-neighbor classifiers can be divided into two categories: fast nearest-neighbor searching [20, 21] and prototype optimization [22, 23].

From the statistical point of view, for pattern classification, a pattern should be assigned to the class having the largest *a posteriori* probability. In this way, the classification error is referred to as the *Bayes error.* The *Bayesian decision rule* is designed to achieve classification with the Bayes error. In general, the Bayesian decision rule uses Bayes' theorem to estimate *a posteriori* probability in terms of class *a priori* and class conditional probabilities. Independence between features [24] is often assumed, so that joint probability can be decomposed into several independent ones. For the sake of simplicity, the *a priori* probability of different classes are assumed equal, and can be eliminated from the computation. Some attempt [25, 26] has been made to estimate the class conditional probability density function of features, such as Kernel Regression Estimator and Parzen Window Estimator.

Rosenblatt [27] suggested a general approach to the automatic learning of discriminants for pattern classification, the Perceptron rule. The concept of this rule is rather simple and rational. By using measurement features, linear hyperplanes which divide the feature space into different class regions are constructed. If a pattern is not correctly classified into its true class according to the hyperplanes, then the normal vectors of the hyperplanes are changed in the direction tending to make the

pattern classified correctly. In general, the Perceptron rule functions well for linear cases; however, it is incapable of achieving good classification results for nonlinear problems. Therefore, the polynomial discriminant rule has been proposed to serve nonlinear classification. This classification rule makes use of not only the first order but also higher order features. With higher order features, nonlinear hyperplanes such as Hyperbolic Paraboloids can be constructed to divide a feature space. But, because of the polynomial combination of features, the number of features increases rapidly. Accordingly, quadratic discriminants are the most commonly used. To reduce the feature dimension, Schürmann [28] has developed an algorithm which offers a systematic procedure to select the best feature subsets.

In general, by means of statistical properties (such as means and variances of features), statistics-based classification is highly resistant to noise, but structural variation of characters will degrade its performance considerably.

B. Syntactics-Based Classification

Syntactics-based classification is based on concepts from formal language theory [29, 30, 31]; grammars at all levels in the Chomsky hierarchy are used to describe character models. In general, a pattern is decomposed into sub-patterns of pre-defined primitives. These primitives are interpreted as symbols in some grammars, a set of syntactic rules for generating sentences from the given symbols. For characters, the primitives contain horizontal, vertical and slant strokes, loops, curves, T-joints, end

points, and so on. With a grammar, pattern classification has the same physical meaning as parsing a sentence and deciding whether it is recognizable or not. If the grammar is simple, the parsing procedure can be designed to become a tree-like structure such as a decision tree; otherwise, some inference engine may need to be applied, just as in AI. For syntactics-based classification, the most challenging problem is how to construct the grammars or decision trees automatically. Research effort can be seen in [32, 33].

Dynamic Programming (DP) is another syntactics-based classification method. DP deals with two strings and computes the distance between them by finding an *optimal assignment* or *correspondence* between the elements of one string and those of the other. When computing the distance, DP allows *substitution*, *insertion* and *deletion* of elements so that the two strings can have different lengths and different order of elements. Another advantage of DP is that it can derive both the optimal assignment and the resulting distance between two strings at the same time. Recently, Yamada [34] has been active in using DP for both character and map recognition

Basically, syntactics-based classification performs robustly against structural variation, but is sensitive to noise (such as the breaking of strokes, and the closing and opening of loops).

C. Neural-Network-Based Classification

Neural network models have many different names, such as connectionist models, parallel distributed processing models, and neuromorphic systems [35]. Originally, they were *inspired* by man's understanding of the brain; but with a broadened scope, they are not necessarily conformed strictly to that understanding. In general, a pattern classification approach is called a neural network if it contains three basic components:

- a set of computation neurons (or nodes):
- a regular structure (*topology* and *interconnection*) among computational neurons;
- an automatic learning rule (often derived from the optimization of the predefined *cost* functions).

Simply speaking, neural networks automatically adapt themselves to the required processing capacity, using large numbers of simple processing elements operating in parallel.

Neural networks provide a great degree of robustness or fault tolerance because they make use of many processing units, each with primarily local connections. Damage to a few nodes or links thus will not cause a significant degeneration in the overall performance. In general, neural network models are *non-parametric* and make weak

assumptions, so that they may be applied without much trouble from one data set to another. Among various neural net models, multi-layer perceptrons with back-propagation error correction are most commonly used in pattern recognition applications. The back-propagation learning rule can modify the initial interconnection weights in the direction of reducing the classification error. Other well-known and representative neural network models include Adaline and Madaline [36], Bidirectional Associative Memory (BAM) [37], the Hopfield model [38], the Boltzmann machine [39], Simulated Annealing [40], Counterpropagation [41], Self-Organizing Map [42], Adaptive Resonance Theory (ART) [43], Neocognitron [44], Radial Basis Function (RBF) [45], and so on.

1.2 The Demand for Multiple Experts

In the early days of pattern recognition, a lot of research was focused on character recognition. One reason was that characters were handy to deal with, and were regarded as a problem which could be solved easily. However, when the research advanced from printed into handwritten character recognition, a great deal of challenge in solving this problem surfaced, such as unconstrained shape variations, different writing styles, different kinds of noise which may break the strokes in the characters or change their topology, and so on. Even so, many researchers still continue to develop and implement algorithms for recognizing totally unconstrained handwritten

characters, expecting that an OCR machine with a high recognition rate and a zero substitution rate can be achieved. However, because the problem is intrinsically too complicated, it is impossible to predict when and how a satisfactory solution may appear.

1.2.1 Teamwork – A Human Approach to Resolve Difficult Problems

Since difficulties have seriously and continuously impeded the successful development of practical OCR machines, it is natural and useful to consider the approaches that humans may adopt to tackle a complicated problem. In a human society (*e.g.* a company), when there is a job which is considered to be very difficult, then instead of assigning one person to take care of it, we tend to assign it to a team of several people. Experience tells us that when each member of this team can work together effectively, then usually a better solution can be found. No wonder that there are so many sayings in different languages recommending the advantage of team work, such as "*two heads are better than one*". All of them describe a simple belief that *group decision is better than any individual's.*

1.2.2 Basic Requirements for Teamwork

However, belief in teamwork is not always valid unless three basic requirements are satisfied: first, there must exist several qualified individuals who are suitable to

serve this job; secondly, the individuals are willing to work together and have a successful channel to communicate with each other; and thirdly, there exists an efficient and effective mechanism to resolve the potential conflicts among the individuals, and to aggregate their various opinions into a final consensus. For example, suppose the job is to create a new generation of apple which can be cultivated under warm temperatures (about 30^0 C). It will be impossible to organize a successful team if no one has the biology or genetics background. Furthermore, even when there are qualified candidates, if they are not willing to cooperate as a team (which happens to human experts), or they speak different languages (such as one speaks only English, and the other one only French) without translation, a useful solution is not achievable. Finally, it is obvious that even if the first two conditions are satisfied, a sound and valuable solution can only be found when there exists a good decision mechanism which can aggregate individual's opinions by strengthening their expertise and avoiding their weaknesses.

1.3 Combination of Multiple Classifiers in OCR

Based on the above discussion, the concept of teamwork can also be applied to the recognition of unconstrained handwritten characters. Pioneered by Suen [46], a new direction in the OCR research field has emerged, *viz* "Combination of Multiple Experts (CME)". For character recognition, a *classifier* is called an "*expert*" when it

has attained an expert-like performance in recognizing characters. Here, we use the term "expert" to emphasize that the combined classifiers should not behave poorly. This requirement indeed corresponds to the first condition of a successful team in the above discussion. Theoretically, CME for character recognition is based on the idea that *classifiers with different methodologies or different features are often complementary to each other; hence, the combination of different and complementary classifiers may reduce errors considerably and achieve a higher performance accuracy*, just as the decision of a panel of human experts is usually superior to that of a single individual. Many such approaches have now been developed, and have already shown encouraging results [46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57], which reveal that CME is a promising avenue to achieve the expected goal mentioned above.

1.3.0.1 Hardware Concerns in CME

However, CME is not achieved without paying a price. In fact, *cost* is the most serious constraint for using multiple classifiers. It means that not only a longer computation time will be spent to recognize a pattern, but also bigger computer hardware is required to support it. Fortunately, looking at the history of computer hardware development, computer hardware and design technologies have steadily and continuously made very rapid advances, such as the introduction of high-speed CPU processors, large memory capacities of RAM, sub-micro VLSI design, parallel architectures, and so on. This indicates that our ability to design a powerful yet low-cost piece of

hardware with a large memory and high computing speed has improved. As a result, the cost constraint is no longer unsolvable, and can finally be overcome. Accordingly, we may spend most of our effort on software aspects for achieving a high recognition rate, without hardware considerations imposing much of a constraint.

1.4 Different Architectures of CME

A conventional recognition system consists of only a single classifier; the output of the classifier is the same as the system output. However, several classifiers are involved in a multi-classifier recognition system; their outputs will be further processed in order to derive the final decisions. In general, there are three kinds of CME architectures, which differ in how and when the classification outputs of individual classifiers are processed.

A. Integrated CME

Integrated CME means that each classifier takes charge of a sub-task of recognition; the overall recognition will be accomplished by cooperation among all classifiers involved. For instance, one classifier may be designed specifically to recognize loops, and another for horizontal or vertical lines. Some classifiers may even be good at recognizing a certain character. During recognition, individual classifiers will be dynamically triggered to function whenever they can provide useful information to the

overall recognition. With this architecture, it is important to have efficient coordinating systems, which can interactively supervise and control all recognition and information between the whole system and individual classifiers. For example, *Blackboard system* [58, 59] is a commonly-used architecture.

B. Sequential CME

Sequential CME means that there are different process priorities for individual classifiers. The classifier with a higher priority will be triggered earlier than the one with a low priority. Each classifier functions as a sifter, which reduces the range of possible candidates for the input pattern. Ideally, the last classifier should produce one and only one candidate, which is the character most similar to the input pattern. Figure 2 shows the block diagram of this architecture, which contains K classifiers $e_1, \cdots, e_k, \cdots$, and e_K. Obviously, except for the first classifier e_1, classifier e_k will begin to process only when classifier e_{k-1} has finished its recognition, and the output of classifier e_K then becomes the final system output. *Multistage* [60] and *Multilevel* [61] recognition are two commonly-used methods in sequential CME.

C. Parallel CME

Parallel CME means that all classifiers have the same process priority. Therefore, combination will take place only after all classifiers have finished their recognition processes. Figure 3 shows the block diagram of parallel CME. Obviously, the outputs of individual classifiers become the input of a combination module (call it E), whose

Figure 2: Block diagram of sequential CME.

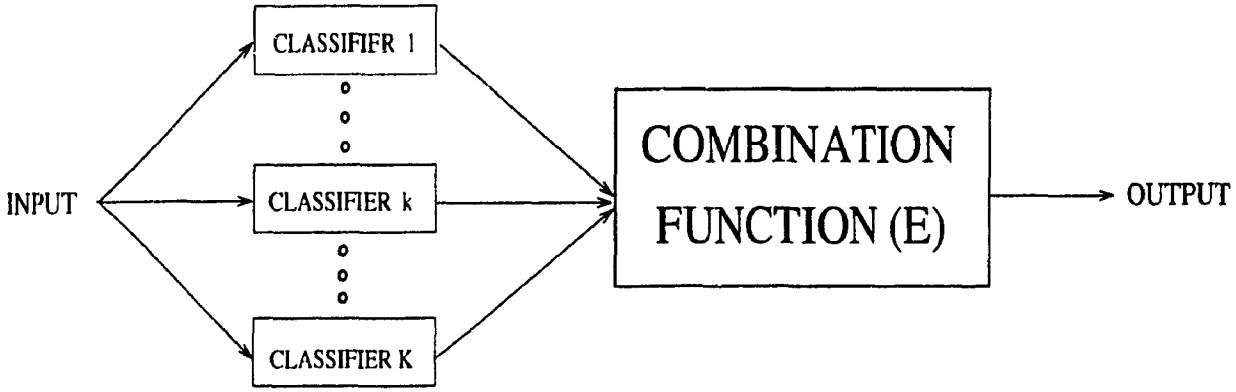


Figure 3: Block diagram of parallel CME.

output then forms the final decision of the system. Thus, the goal of CME research is to ascertain that the combination module E will produce the best recognition performance.

Integrated CME may involve dynamic control and uncertainty management which are quite challenging. But, because most of the existing character classification procedures perform complete recognition process, it is unsuitable to combine them by using the structure of integrated CME. As for sequential and parallel CME, Shridhar *et al.* [62] have performed a series of experiments and made comparisons between

them. They gave two conclusions: (1) For sequential CME, applying the classifier with lower error rate first, and higher ones later, is better than applying them in the reverse order; and (2) parallel CME usually achieves less error than the corresponding sequential CME. It is easy to realize that sequential CME has a built-in drawback, *error accumulation*, which does not appear in parallel CME. As a result, parallel CME can outperform sequential CME. Concerning the availability of classifiers and the recognition accuracy of CME, this thesis focuses mainly on parallel CME; in the remaining discussion of this thesis CME stands for parallel CME.

A similar approach to CME is to collect a set of distinctive features, and develop a single but powerful classifier using all of them. However, there are several disadvantages which make us believe that CME is better than this approach. First, distinctive features may be represented in very diversified forms, *e.g.*, they may be continuous variables, binary values, discrete labels, structure primitives, and so on. Second, different features may have different physical meanings and different scales, *e.g.*, some feature represents "*length*" or "*size*", and another "*type of primitives*" or "*relation* among primitives". Third, the dimension of the collected distinctive features in general tends to be large. Due to the undesired phenomenon - *the curse of dimensionality* [63], a classifier designed by a large number of features often performs well on the training data set, but poorly on the unseen testing data set. Fourth,

there are no clear guidelines to design such a classifier. In other words, the complexity of designing such a classifier is too high to be practically realized. However, the research for designing a highly discriminant classifiers is still a very important topic because classifiers are the basic components of CME. Any improvement of the individual classifiers can finally benefit the recognition performance of CME.

1.5 Abstract-Level and Measurement-Level CME

Since the outputs of classifiers are the basic information which will be fed to the combination module E, it is therefore important to analyze what kinds of output information various classifiers can support. It is believed that different kinds of output may need different combination functions to fully and best explore their expertise. From our observation, the information output by various classifiers can be divided into two levels[1]:

(1) The *abstract* level: a classifier outputs only an unique class label, or a subset of class labels when it cannot decide on the identity of a confusing pattern.

(2) The *measurement* level: a classifier assigns each class label a measurement value to indicate the possibility that the input pattern belongs to the class.

[1]Some researchers [54] consider that classifiers supply three levels of information, *i.e.* the *abstract*, *rank*, and *measurement* levels. Because rank-level information is derived from measurement-level information, without loss of generality the authors consider that it is appropriate to divide the output information into two levels only.

Obviously, the measurement level contains richer information than the abstract level. In fact, abstract-level information can be easily derived from measurement-level information through an *information reduction* or *abstraction* process. Classifiers producing information at the abstract or measurement levels are called *abstract-level* or *measurement-level classifiers*. The combination of abstract-level or measurement-level classifiers is called *abstract-level* or *measurement-level CME*. In general, if the combined classifiers are measurement-level classifiers, measurement-level CME will produce a better recognition accuracy than abstract-level CME. But, it is by no means true that abstract-level CME is not important. In fact, some classifiers, such as various kinds of decision trees, can only produce abstract-level information. Since every classifier is able to offer abstract-level information but not every classifier can offer measurement-level CME, abstract-level CME indeed has the broader applicability.

1.6 Research Related to CME

Because of the generality and effectiveness of the notion, ideas similar to CME can be found in many other fields, such as social choice in election systems [64, 65, 66], consensus of decisions [67, 68], sensor fusion [69], disease diagnosis of medical systems [70, 71, 72], and so on. Many methods have already been implemented from previous studies, *e.g.* majority voting [46, 55], Borda count [73, 74], the evidential theory of

Dempster-Shafer [71, 75, 76], the certainty factor model of MYCIN [77], probability of Bayesian [54, 78] or Prospector [79], and fuzzy logic [80]. However, although these methods are designed from different points of view, almost all of them have one feature in common, that is they require an assumption of either total independence or maximal dependence in the behavior of the considered experts. Both assumptions actually constrain the practicality of these methods, because they are seldom true in real applications. Not surprisingly, in applying them to character recognition, the performance of the recognition system may deteriorate considerably when these assumptions are not satisfied.

1.7 Objectives and Organization of This Thesis

It appears that the study of the combination problem is now only at its preliminary stage. For character recognition, although several approaches have been developed, more effort should be devoted to theoretical issues such as, "Is it possible to prove that a CME function achieves the optimal recognition performance?", or "Is it possible to derive advantageous properties from a CME function theoretically?" The primary goal of this research is to obtain effective combination approaches for a multiple classifier system that takes advantage of the individual strengths of abstract-level or measurement-level classifiers. Due to the two different levels of information supported by various classifiers, we believe that at least two CME functions should

be developed, one for abstract-level classifiers and the other for measurement-level classifiers. Besides focusing on recognition accuracy, other CME related issues are addressed as well, such as

- What are the advantages and disadvantages of these functions?
- Can the disadvantages of the functions be alleviated, and how?
- When several classifiers are available, how can a subset of classifiers be selected to produce the highest recognition accuracy among all subsets containing the same number of classifiers?
- Is it true that combining more classifiers can always produce an equal or even better recognition performance?
- Given the required performance for recognition accuracy and error limitation of an application, how can a combination procedure adapt its performance to the required one automatically?
- If CME can achieve better performance than individual classifiers, it seems to be reasonable to assume that even better performance can be achieved by combining both different CME approaches and the original individual classifiers. This new combination scheme is called *consecutive* CME. Then, is it true that consecutive CME will always produce better recognition accuracy?

In this thesis, we first make a simple review of the existing CME methods, then propose our combination methods, and describe the comparisons of the experimental results of different combination functions. In total, this thesis contains nine chapters. Chapter 2 lists and explains the basic terminology used in both character recognition and CME, so that the readers can understand the terms more easily when they are mentioned in the later discussion. Formal definitions of abstract-level and measurement-level CME are also specified in this chapter.

Chapter 3 reviews the existing and representative classifier combining functions of abstract-level and measurement-level CME, respectively. For abstract-level CME, this contains the majority voting principle, Bayesian combination approach, evidence aggregation by Dempster-Shafer theory, and associative switch. For measurement-level CME, it contains Borda count, rank reordering with set intersection and union, Dempster-Shafer approach, and polynomial classifiers. These approaches reveal that CME can be studied from different methodologies and different points of view.

In Chapter 4, a novel approach for abstract-level CME is proposed, the "Behavior-Knowledge Space Method" (the BKS method). First, we define a behavior knowledge space. Then we argue that a behavior knowledge space can concurrently record the abstract-level classification decisions of individual classifiers to an input pattern. Because of concurrent recording, this method does not require the classifier-independence assumption, which is common in other CME methods. With this

method, several CME related issues are discussed theoretically, such as "Does it perform better than other CME methods?", "Can consecutive CME improve its recognition performance?", "Is it true that combining more classifiers will always produce an equal or even better recognition performance?", and "What are the intrinsic constraints in this method, and how can they be avoided?" Then, an extension to combine multiple classifiers which produce more than one abstract-level classification decision is discussed. Finally, after the combination stage, a possible direction to improve recognition performance is pointed out.

In Chapter 5, three approaches for measurement-level CME are proposed. The first one is the Linear Confidence Aggregation method (LCA), which first transforms measurement values supported by individual classifiers into probability-based confidences, then all confidences are summed up linearly. The second approach is the Bayesian Confidence Aggregation method (BCA). Instead of aggregating confidence linearly as LCA, BCA makes use of Bayes' theorem, so that it aggregates confidences by multiplication. The third approach is motivated by a new understanding of CME as intrinsically a pattern recognition problem. Accordingly, neural network techniques can be applied to CME. In this approach, due to different possible physical meanings and scales, measurement values are transformed into a new form of "*likeness*" before being fed to neural networks. The transformation of measurement values enables this approach to satisfy the second requirement for successful teamwork. In the last

part of this chapter, three strategies to improve the performance of the network are proposed, which are *training by boundary patterns, training by partition*, and *weight reduction.*

Based on 46,451 handwritten samples of ITRI's numeral database, a series of experiments have been performed to compare the performances of different CME functions. These experiments are divided into two groups for abstract-level and measurement-level CME. Chapter 6 specifies the design methodologies of these experiments, their experimental results, and some observations drawn from the experiments.

Since CME is in fact a pattern recognition problem, the design techniques of CME can surely be applied to the design of classifiers. In Chapter 7, two new classifier design approaches using the same type of features and the same classification methodology are introduced: *recognition by parts* and *recognition by pair classifiers.* Simply speaking, recognition by parts is to construct several classifiers, each of which does the complete recognition task by using only partial information of the overall character features, then a combination model aggregates the classification results of the classifiers together to produce the overall recognition result. Recognition by pair classifiers is to use pair classifiers, each of which is constructed only based on the features of one pair of classes, then a combination model will aggregate the classification results of all pair classifiers. As a result, the final recognition decision is produced.

Because the basic component of a multiple-classifier system is a classifier and CME turns out to be a generic pattern recognition problem, it is essential to improve the recognition efficiency of classifiers. Chapter 8 describes our effort on the research to derive the optimized prototypes for the nearest neighbor classifier. We proposed a new method using a four-layer network with a novel error function. The derived prototype update rule exhibits a deterministic-annealing property, and we have shown that the famous LVQ2 [81] algorithm can be regarded as a special case of this approach. Through experiments with different data sets, it manifects that this method outperforms the other two prototype optimization methods, *i.e.* Yan's method [82] and LVQ2.

Finally, Chapter 9 concludes this study of CME by summarizing our research results and pointing out future research directions.

Chapter 2

Terminology and Problem Formulation

2.1 Introduction

This chapter serves two purposes: (1) to illustrate the basic terminologies which are closely related to the CME study, and (2) to make a clear and formal specification of CME. Both of them are intended to make this thesis more understandable.

2.2 Basic Terminology

A *class* is a set of samples which are grouped together to serve a common purpose. The criteria for grouping a class may be based on feature similarity or the abstract identity of patterns. For example, in a supermarket, according to their size, eggs are

classified into three classes: small, medium, and large. For another example, numerals are divided into 10 classes, one for each digit. Among the samples of the same numeral class, some may have a dramatically different shape from others; however, they still belong to the same class, because they have the same abstract identity of the class. A *class label* is a code assigned to the corresponding class, which enables an easy referential representation to the class.

A *classifier* is a function which maps a pattern from its feature or image space into an abstract-level or measurement-level space. When a pattern is mapped into an abstract-level space, usually only one class label will be the classification decision; however, when it is mapped into a measurement-level space, corresponding to each class there is a measurement value which indicates the degree of the pattern belonging to the class. For recognition, a mapping is considered to be successful if the first choice of class label made by a classifier is the same as that made by an educated human being. The *ground-truth* class label denotes the class to which a pattern genuinely belongs.

Recognition performance involves the indices for specifying the correctness of the classification of a classifier, generally represented by four items: the recognition, substitution and rejection rates, and reliability. The recognition rate is the ratio of the total number of correctly recognized samples to the total number of testing samples; the substitution rate is the ratio of the total number of incorrectly recognized samples

to the total number of samples. Interchangeably, the substitution rate is called the error rate. When a classifier makes a decisive classification of a pattern, we say that the classifier *recognizes* the pattern. In some cases, a classifier may not be able to recognize a pattern if it does not have enough confidence to make an absolute decision. This happens when the quality of the image is poor or the pattern has several similar references in different classes. When a classifier does not recognize a pattern decisively, we say that the classifier *rejects* the pattern. Accordingly, the rejection rate is the ratio of the total number of rejected samples to the total number of samples. Reliability stands for the probability that the classification decision is correct when a classifier makes a decisive recognition. Therefore, reliability is defined as the ratio of the total number of correctly recognized samples to the total number of both correctly and incorrectly recognized samples. Usually, the recognition, substitution, and rejection rates are expressed by percentages, but reliability is a real value which ranges from 0.0 to 1.0. Obviously, the summation of the recognition, substitution and rejection rates is equal to 100%. Expressed by arithematic forms, the four performance indices are

$$\text{Recognition rate} = \frac{\text{No. of samples correctly recognized}}{\text{Total No. of samples}} * 100\%$$

$$\text{Substitution rate} = \frac{\text{No. of samples incorrectly recognized}}{\text{Total No. of samples}} * 100\%$$

$$\text{Rejection rate} = \frac{\text{No. of samples not recognized}}{\text{Total No. of samples}} * 100\%,$$

and,

$$\text{Reliability} = \frac{\text{No. of samples correctly recognized}}{\text{No. of samples correctly recognized} + \text{No. of samples incorrectly recognized}}.$$

A decision equation represents the criterion based on which a classifier recognizes or rejects a pattern. Generally, the criterion is expressed in terms of belief values, which indicate the degree that a pattern belongs to each class. Here, belief value is a general form which may be expressed by distance, similarity, confidence and so on. For the purpose of rejecting ambiguous patterns, a decision equation often consists of a threshold parameter. Several design schemes [54] for decision equations are commonly used in practice: the first-choice may be simply selected as the classification decision; or if the difference in belief between the first-two-choices is larger than a given threshold, then the first-choice will serve the classification decision, otherwise the pattern is rejected with no recognition.

A confusion matrix (CM) is a two-dimensional table which shows how many samples genuinely belonging to one class are recognized as a certain class. Each classifier maintains its own confusion matrix with discrete values for both the horizontal and vertical axes. For an M-class recognition problem, the columns of a CM stands for the ground-truth class labels of patterns, therefore it contains M values of the M class labels; each row stands for the class label to which a pattern is recognized, therefore it contains $M + 1$ possible classification decisions $\{1, \cdots, M + 1\}$. In the following

CM example with M pattern classes,

	R_1	R_2	$\cdots$	R_j	$\cdots$	R_M	R_{M+1}
C_1	n_{11}	n_{12}	$\cdots$	n_{1j}	$\cdots$	n_{1M}	$n_{1(M+1)}$
C_2	n_{21}	n_{22}	$\cdots$	n_{2j}	$\cdots$	n_{2M}	$n_{2(M+1)}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
C_i	$\vdots$	$\vdots$	$\vdots$	n_{ij}	$\vdots$	$\vdots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
C_M	n_{M1}	n_{M2}	$\cdots$	n_{Mj}	$\cdots$	n_{MM}	$n_{M(M+1)}$

C_i and R_j mean that the ground-truth class label of a pattern is i and the recognized class label is j. n_{ij} represents the number of patterns which belong to class C_i but are recognized as R_j. It is very convenient to derive statistical data from a confusion matrix, such as

the total number of samples belonging to class $i = \sum_{j=1}^{M+1} n_{ij}$,

the total number of samples recognized as class $j = \sum_{i=1}^{M} n_{ij}$,

the total number of samples belonging to class i and correctly recognized $= n_{ii}$,

and,

$P(x \in C_i \mid e(x) = R_j) = n_{ii} / \sum_{m=1}^{M} n_{mj}$

where $e(x)$ stands for the decision of a classifier with respect to a given pattern x.

2.3 Problem Formulation

As described in Chapter 1, two kinds of CME exist: abstract- and measurement-level CME. They are different because they combine two different levels of output information supported by abstract-level and measurement-level classifiers. To formulate the two kinds of CME, their common elements will be described first, then the distinguishing parts will be specified, respectively.

Let S be a pattern space which consists of M sets $S = C_1 \cup \cdots \cup C_M$, each C_i, $i \in \Lambda = \{1, \cdots, M\}$ representing a set of specified patterns called a class (*e.g.* $M = 10$ for numeral recognition). e_k stands for an expert or classifier k where $k = 1, \cdots, K$, and K is the total number of classifiers. x denotes an input pattern.

2.3.1 Formulation of Abstract-Level CME

$e_k(x) = j_k$ means that expert k assigns the input x to class j_k, where $j_k \in \Lambda \cup \{M+1\}$. When $j_k \in \Lambda$, it means that expert k accepts and recognizes x (either correctly or incorrectly); otherwise expert k rejects x. To simplify the notation, $e_k(x)$ is replaced by $e(k)$. Then, the research goal of abstract-level CME becomes, "When K classifiers give their individual classification decisions $e(1), \cdots, e(K)$ about the identity of x, what is the combination function $E(e(1), \cdots, e(K))$ which will produce the final classification decisions effectively?" In fact, this problem can be illustrated more

clearly as

$$\text{given} \left\{ \begin{array}{rcl} e_1(x) & = & j_1 \\ e_2(x) & = & j_2 \\ & \vdots & \\ e_K(x) & = & j_K \end{array} \right\} \xrightarrow{?} E(x) = j$$

where $E(x)$ is a combination function of the multiple classifiers which gives x one definitive class label $j \in \Lambda \cup \{M+1\}$. Expectantly, the accuracy of decision j is higher than that of decision j_k, $\forall j_k \in \{1, \cdots, K\}$.

2.3.2 Formulation of Measurement-Level CME

$e_k(x) = \{m_k^i(x) | \ \forall i \ (1 \leq i \leq M)\}$ means that expert k assigns the input x to each class i with a measurement value $m_k^i(x)$. With a similar realization as in abstract-level CME, $m_k^i(x)$ is expressed as m_k^i to simplify the notation. Thus, this problem can be formulated as

$$\text{given} \left\{ \begin{array}{rcl} e_1(x) & = & m_1^1, \cdots, m_1^M \\ e_2(x) & = & m_2^1, \cdots, m_2^{K'} \\ & \vdots & \\ e_K(x) & = & m_K^1, \cdots, m_K^M \end{array} \right\} \xrightarrow{?} E(x) = j$$

where $E(x)$ is the combination function which gives x one definitive class $j \in \Lambda \cup \{M+1\}$.

Chapter 3

Previous Studies on CME

3.1 Introduction

Motivated by different points of view, many CME approaches have been explored. As a result, several combination methods for both abstract-level and measurement-level CME have been developed from previous studies. Although they have shown very promising recognition performance, each of them more or less endures its own built-in constraints, such as lacking learning ability, requiring that classifiers behave independently of each other, and so on. Their effectiveness may be considerably reduced when some constraints are not fulfilled in practical applications. In this chapter, the methods of abstract-level and measurement-level CME will be introduced, respectively. Advantages and disadvantages of them will also be discussed.

3.2 Methods of Abstract-Level CME

Four representative methods of previous studies on CME will be described in this section: majority vote [46, 51, 52, 55], Bayesian principle [54, 78], evidence aggregation of Dempster-Shafer (D-S) theorem [71, 72, 75, 76], and associative switch [83].

Simply speaking, voting is a common democratic approach based on "the opinion of the majority wins". It treats classifiers equally, without considering their differences in performance. Therefore, it has no learning ability, and will not improve its performance even when more and more data are accumulated. The Bayesian approach uses the Bayesian formula to integrate classifiers' decisions; usually it requires an independence assumption in order to tackle the computation of the joint probability. The D-S formula, which has frequently been applied to deal with uncertainty management and incomplete reasoning, can aggregate committed, uncommitted and ignorant beliefs. It allows one to attribute belief to subsets, as well as to individual elements of the hypothesis set. An advantage of this approach is its ability to model the *narrowing* of the hypothesis set with accumulation of evidence. Both Bayesian and D-S approaches make use of probability to describe the different qualities of classifiers' decisions. However, in the Bayesian approach, the sum of $P(C)$ and $P(\sim C)$ is equal to one where $P(C)$ represents the probability that C is true; this is not necessarily true for the D-S approach. Associative switch is motivated from an idea which is quite different from the other three methods. Instead of making use of the

classification decisions of all K classifiers, associative switch only chooses a single classifier's decision as the final decision. The key task of this method is to find a mechanism which can predict which classifier will produce the correct identity of the current input pattern efficiently and effectively.

The following three sub-sections (3.2.1 to 3.2.4) will briefly introduce these three methods. In sub-section 3.2.5, several interesting conclusions and further research topics drawn by Xu *et al.* [54] will be mentioned.

3.2.1 Majority Vote

This method exists commonly in real-life activities such as various elections, community decisions and court verdict. It means "the opinion of the majority wins", *i.e.* the opinion which gets the largest number of votes wins. For counting votes, $v_k(i)$ is defined as the individual voting function, with a value 1 when expert k votes x for class i, otherwise 0; $BEL(i)$ is defined as the global voting function which counts the total number of experts who vote x for class i. Formally, the two symbols can be expressed as

$$
\begin{aligned}
v_k(i) &= \text{the individual voting function,} \\
&= \begin{cases} 1 & \text{, when } e_k(x) = i \text{ and } i \in \Lambda, \\ 0 & \text{, otherwise;} \end{cases}
\end{aligned}
$$

and,

$$BEL(i) = \text{the global voting function,}$$
$$= \sum_{k=1}^{K} v_k(i).$$

The main advantages of this method are its easy implementation, fast computation, small memory requirement, and no demand for learning. The major disadvantage is that classifiers are treated with equal weight, without considering the different qualities of their behavior. It is not capable of learning, and is not able to improve its performance as more data are processed.

3.2.2 Combination by Bayesian Formula

Different from the voting method, the Baysian method accepts error-embedded behavior of individual classifiers by combining their conditional probabilities. The conditional probability is expressed as $P(x \in C_i \mid e_k = j)$, which means the probability that input x belongs to class i when classifier k assigns x to class j. $P(x \in C_i \mid e_k = j)$ can be derived from the confusion matrix of classifier k, which records the classification behavior on training samples by classifier k.

In general, when each classifier gives its classification decision to an input pattern x, the belief value that x belongs to class i is computed from a belief function given by

$$BEL(i) = BEL(x \in C_i \mid e_1(x), \cdots, e_K(x)). \tag{1}$$

If each classifier k offers its decision as j_k, then $BEL(i)$ can be expressed by a conditional probability as

$$BEL(i) = P(x \in C_i \mid e_1(x) = j_1, \cdots, e_K(x) = j_K, EN^K) \tag{2}$$

where $i \in \Lambda$ and EN^K denotes the classification environment generated from combining the K classifiers. The purpose of EN^K is to identify under which environment the conditional probability is computed. In fact, it explicitly specifies that the conditional probability should be computed from the knowledge space which is constructed by exactly the K classifiers. With the assumption that classifiers are independent of each other, by using the Bayes formula, Equation (2) can be decomposed and simplified as

$$BEL(i) = \eta \prod_{k=1}^{K} P(x \in C_i \mid e_k(x) = j_k, EN_1^k) \tag{3}$$

where η is a normalization coefficient which makes $\sum_{i=1}^{M} BEL(i) = 1$, and EN_1^k is the classification environment generated from the classifier k alone, which is indeed the confusion matrix of classifier k.

3.2.3 Evidential Aggregation by Dempster-Shafer Theorem

The Bayesian approach assumes that the commitment of belief to a hypothesis implies commitment of the remaining belief to its negation, *i.e.* the assumption that belief in A is equivalent to $P(A)$ so that the resulting belief in NOT A is $1 - P(A)$. This assumption is not always valid, because evidence partially in favor of a hypothesis

is not construed as evidence partially against the same hypothesis. In fact, human experts often give their opinions conservatively, meaning that when an expert offers his support to set A with a certain belief m, the remaining unassigned belief 1 - m may just be reserved for the universal set containing all single elements of decision, and not necessarily for the negation for A as would be assumed in the Bayesian model. To serve this and other purposes, Dempster and Shafer proposed a mathematical theory, called the Dempster-Shafer theory [76], to aggregate evidence. This theory not only avoids the above Bayesian assumption, but also allows one to attribute belief to subsets, as well as to individual elements of the hypothesis set. One other important property of this model is its ability to narrow the hypothesis set with accumulation of evidence. In other words, the hypothesis set originally contains all individual elements: as more pieces of evidence are collected, the number of the possible individual elements will be reduced. Due to these distinguishing properties, Dempster-Shafer theory has been broadly applied in uncertain reasoning to represent and manipulate incomplete and imperfect knowledge.

In the Dempster-Shafer theorem, $\Theta = \{1, 2, \cdots, M\}$ denotes an exhaustive and mutually exclusive universal set. Θ is called the *frame of discernment.* A function $m : 2^{\Theta} \rightarrow [0, 1]$ is called a *basic probability assignment* (*bpa*) if it satisfies $m(\emptyset) = 0$ and $\sum_{A \subseteq \Theta} m(A) = 1$. The quantity $m(A)$ represents our exact belief in the proposition represented by set A. Let there are K evidences, and $m_1, \cdots, m_K$ denote the

corresponding K *bpa*s respectively; each *bpa* represents an evidence. The Dempster-Shafer rule defines a new *bpa* $m = m_1 \oplus \cdots \oplus m_K$ by aggregating these K evidences $m_1, \cdots, m_K$ from the following formula:

$$\begin{aligned} m(A) &= m_1 \oplus m_2 \oplus \cdots \oplus m_K(A) \\ &= \kappa \sum_{X_1 \cap \cdots \cap X_K = A} \prod_{i=1}^{K} m_i(X_i) \end{aligned}$$

and

$$\begin{aligned} \kappa^{-1} &= 1 - \sum_{X_1 \cap \cdots \cap X_K = \phi} \prod_{i=1}^{K} m_i(X_i) \\ &= \sum_{X_1 \cap \cdots \cap X_K \neq \phi} \prod_{i=1}^{K} m_i(X_i). \end{aligned}$$

where $A, X_1, \cdots, X_K \subset \Theta$ and $\log \kappa$ is called the weight of conflict, which specifies a conflict index among the K evidences.

Finally, the belief function that input x belongs to subset A is

$$Bel(A) = \sum_{B \subseteq A} m(B).$$

Since the goal is to find which class gets the largest belief value, only the belief values of single-element subsets need to be computed, and the class with the largest belief value is selected as the final decision.

3.2.4 Associative Switch

One different point of view about CME is that for a pattern x, instead of combining the classification decisions of all classifiers together, the decision of only one classifier is chosen as the final decision. The chosen classifier is expected to have the best classification decision to recognize x among all classifiers. Motivated by this idea, a

method called associative switch for combining multiple classifiers is proposed by Xu *et al.* [83]. As shown in Fig. 4, the switch consists of: (1) a number of knobs which gate the output channels of individual classifiers, and (2) a multi-layer perceptron neural net trained by a backpropagation-like technique. When an unlabeled pattern is input to each individual classifier, it also enters a neural net for associatively recalling a code which controls the knobs to decide whether the output of each classifier should pass through as the final result. The array consists of K knobs $sw_k, k = 1, \cdots, K$, with each sw_k installed on the output channel of classifier e_k, to decide whether or not it is selected as the expert for the present input (*i.e.*, to let its output pass through). So, the output of each knob is given by

$$j_k' = \begin{cases} j_k & \text{, if } o_k \geq o_t, \\ \Phi & \text{, otherwise;} \end{cases}$$

where o_t is a predefined threshold, and o_k is the kth output of the associative controller. When $j_k' = \Phi$, it means that the decision of expert k is not selected. Since the purpose of the associative controller is to recall which classifier has produced the best decision in the context that the input pattern is x, therefore it could be implemented by any existing neural net of heteroassociative memory type [42]. Each training sample contains the information: K classification decisions of K individual classifiers, and the classifiers which make the right decisions. The idea is to have a good mapping from the current K classification decisions of individual classifiers to the best classifier which has the highest probability to make the right classification

decision. Obviously, the input codes to the controller net are the classification result $e_k(x), k = 1, \cdots, K$. Accordingly, the key task is the design of the desired output codes o_k^d, $1 \leq k \leq K$, to train the controller net on a training data set. Once the desired output codes are obtained, the training procedure is just a supervised learning process which will train the controller net automatically. Assume $I(x)$ represents the ground-true class label of pattern x. Xu *et al.* give a simple way to produce the desired output code according to three different cases:

Case 1. If $j_k \neq I(x)$ for all $k = 1, \cdots, K$ (*i.e.*, there is no individual classifier giving the right classification), then let $o_k^d(x) = 0, k = 1, \cdots, K$.

Case 2. If there is only one k such that $j_k = I(x)$ (*i.e.*, there is only one individual classifier giving the right result), then for $k = 1, \cdots, K$, let

$$o_k^d(x) = \begin{cases} 1 & , \text{ for } j_k = I(x), \\ 0 & , \text{ otherwise.} \end{cases}$$

Case 3. When there are more than one individual classifier giving the right classification result (*i.e.*, there is a subset $S_K \subseteq \{1, \cdots, K\}$ for each $k' \in S_k, j_{k'} = I(x)$). In this case, we arbitrarily or randomly choose one k' among S_K and for $j = 1, \cdots, K$, let

$$o_k^d(x) = \begin{cases} 1 & , \text{ for } j_{k'} = I(x), \\ 0 & , \text{ otherwise.} \end{cases}$$

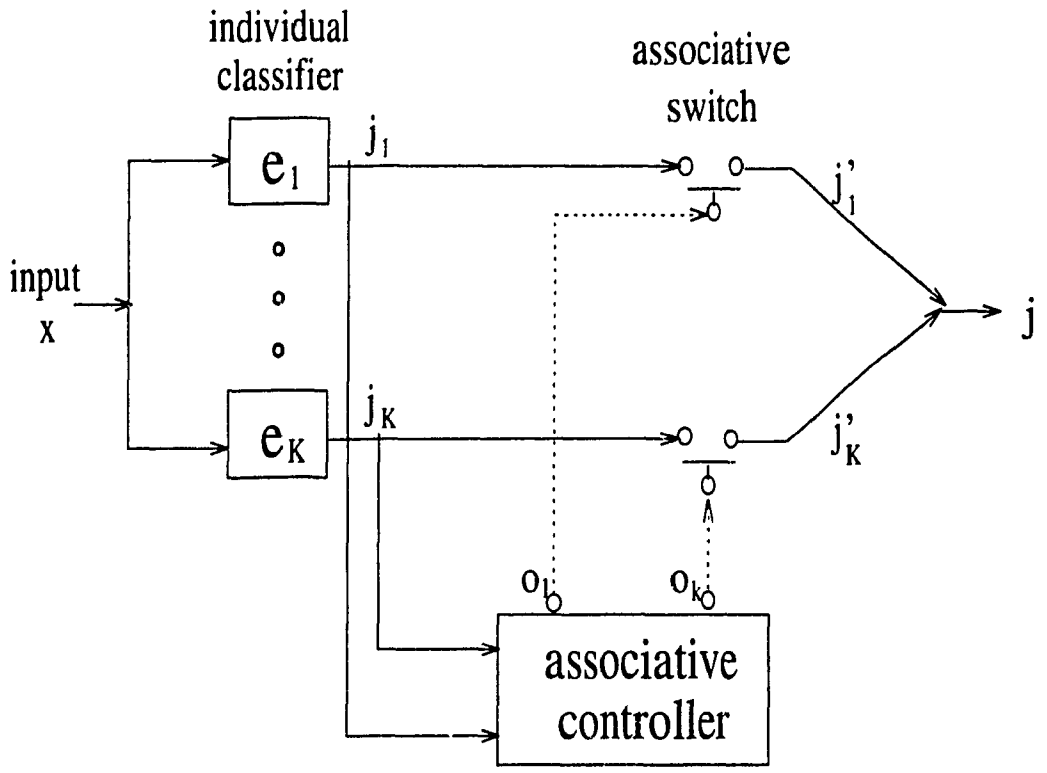


Figure 4: The basic model of associative switch for combining multiple classifiers.

The idea to use a gating network to choose the most suitable single classifier for the current input pattern also appeared in other articles, such as [84] and [85].

3.2.5 Comparison of the Abstract-Level CME Methods

Xu *et al.* [54] performed two experiments with different sizes of learning samples on the above three methods (voting, Bayesian, and Dempster-Shafer), and drew four conclusions:

1. If the confusion matrices of individual classifiers are well learned, then the Bayesian method performs best; otherwise, it will degenerate rapidly.

2. The Dempster-Shafer method is very robust in every situation.

3. Both the Dempster-Shafer method and the voting method behave well, and the Dempster-Shafer method is better than the voting method.

4. The combination of multiple experts can make high quality decisions.

They also pointed out the issues which call for further research, *viz.*

1. The methods described are based on the assumption that individual classifiers are independent of each other. How can one generalize these methods or develop a new approach to combine dependent classifiers?

2. Given a set of classifiers, how many classifiers, and which of them can perform efficiently?

3. Can the performance of the combination of multiple classifiers be analyzed theoretically instead of experimentally?

4. Can a method adapt itself to a required performance?

Although the associative switch approach presents a novel concept to combine multiple classifiers, its performance, in general, is not better than the other three methods. Interested readers can compare references [54] and [83] to justify this statement. The

main reason is that the output information of all classifiers is not fully utilized, because it only takes one classifier's result as the final result. As a matter of fact, this is a kind of *loose cooperation* among the classifiers. This observation enables us to argue that for the multiple-classifier systems all classification results should be aggregated together in order to achieve the best recognition performance, which is called *strong cooperation* among individual classifiers.

In fact, collecting data for handwritten numeral recognition is much easier than that for many other applications such as Chinese character recognition or medical diagnosis. As a result, it is possible to collect a numeral data set with huge samples to generate representative confusion matrices. This reveals that for the application of handwritten numeral recognition the derived probabilities of classifiers should be utilized as much as possible.

3.3 Methods of Measurement-Level CME

Four methods of measurement-level CME will be introduced in this sub-section: Borda Count [73, 74, 86], Set Union or Intersection [86], Dempster-Shafer [48, 87], and Polynomial Classifier [88].

3.3.1 Borda Count

When all classifiers give their support to the individual classes with different preferences, one simple and useful way is to group these opinions together by using

the Borda count function to compute the overall ranking of classes. The Borda count function is a group consensus function well-known in the field of multi-person decision making. In fact, it is a generalization of the majority vote. For any particular class i, the Borda count is the sum of the number of classes ranked below i by every classifier. For example, assume $M = 10$ and set $\{1, 7, 9\}$ is output by a classifier, and class 1 is the first choice of decision, 7 the second and 9 the third; then the Borda count of class 1 is 9, of class 7 is 8, of class 9 is 7, and 0 for the remaining classes. If a candidate subset is selected from the set of allowable classes in computing this count, only the classes in this subset will be considered. The definition of this method is given as follows:

For any class i in a candidate subset U, let $B_k(i)$ be the number of classes in U which are ranked below class i. $B_k(i)$ is zero if $i \notin U$. The Borda count for class i is B_count$(i) = \sum_{k=1}^{K} B_k(i)$. The final ranking is given by arranging the class labels included in the union so that their Borda counts are in descending order. Obviously, this count is dependent on the agreement among the classifiers, and it satisfies our intuition that if class i is ranked near the top by more classifiers, its Borda count tends to be larger.

3.3.2 The Intersection and Union Approaches

Heuristically, the purpose of accumulating more classification information is to have a better ability to reduce the size of a candidate class set in which each element

(*i.e.* a class) has a high probability of pertaining to the input pattern. Naturally, different set-reduction operations can be applied to serve this end by reducing the number of classes in the candidate class set without losing the true class. The criteria for success are therefore twofold: The size of the result set should be minimized, and the probability of the inclusion of the true class should be maximized.

Let us refer to the classes ranked near the top as the *neighborhood* of the true class. The objective here is to produce a neighborhood that contains the true class. Ho [86] has developed two kinds of set operation to serve this purpose: *set intersection* and *set union*. In both operations, two stages are performed: first, a candidate set of classes is produced; second, a group consensus function is then applied to re-rank the classes in the candidate set.

In the first approach, a large neighborhood is obtained from each classifier, and the intersection of these neighborhoods is then output as a set of candidates. Hence a class will be output if and only if it is in all the neighborhoods. In other words, a decision is made if it is confirmed by all classifiers. In the second approach, a small neighborhood is obtained from each classifier. The union of these neighborhoods is output as a set of candidates. A class is output as a candidate only if it is in at least one of the neighborhoods.

The candidate classes contained in the set derived by either the unior or the intersection approaches should be further processed to obtain their final ranking. Many

methods can be applied to re-rank their orders, such as the highest rank method, the Borda count method, or logistic regression which uses a logistic function to predict the importance of the rank orders of each classifier's opinion with respect to classes.

3.3.3 Dempster-Shafer Approach

Mandler and Schüermann [48] realized that it is important to utilize all measurement-level information when dealing with CME. However, they noticed that there exist two problems in using measurement-level information. The first is how to reflect the degree of correctness when a classifier gives a measurement value to a class, and the second is how to aggregate all information together on the basis of the degree of correctness. With these two purposes, they propose a procedure to combine multiple classifiers on measurement level. This procedure consists of three steps: first, each measurement value is transformed into a reasonable confidence-like value, responsible only for one class; second, the Dempster-Shafer theory of evidence is used to combine the basic probability assignments to form the evidence function of a single classifier; third, the basic probability assignment of the different classifiers are combined according to the same theory.

3.3.4 Polynomial Classifier

Franke [89] uses a polynomial classifier to derive the final decision. As well as the first-order items of the classifiers' output, their higher-order items are also input to

the polynomial classifier. Suppose $f(x)$ denotes all the input features (including both the first-order and higher-order[1] items), A is the transformation matrix between $f(x)$ and the output y, where $y = A^T f(x)$, and $d(x)$ is the desired classification output. Then the problem becomes "What is the best A (denoted as A^*) which can produce the minimum error between $d(x)$ and $y(x)$ for all learning samples?" Let $C(A)$ be an error function which is defined as

$$\begin{aligned} C(A) &= E[|d(x) - y(x)|^2] \\ &= E[|d(x) - A^T f(x)|^2]. \end{aligned}$$

Therefore,

$$\begin{aligned} C(A^*) &= \min C(A) \\ &= \min E[|d(x) - A^T f(x)|^2]. \end{aligned}$$

In fact, by using the simple mathematics of linear algebra, A^* can be obtained as

$$A^* = E[\{f(x)f(x)^T\}^{-1} E[f(x)d(x)^T].$$

3.3.5 Comparison of the Measurement-Level CME Methods

The advantage of the Borda count is its simplicity: its easy implementation, fast computation, small required memory, and no demand for learning. Its major disadvantages are twofold: first it assumes additive independence among the contribution of individual classifiers; and it treats individual classifiers equally, without considering

[1] For an $n * 1$ feature vector $f = [x_1, \cdots, x_n]$, a family of features can be constructed as $\{\pi_{i=1}^{p} y_i \mid y_i \in \{x_1, \cdots, x_n\}\}$. When $p = 1$, it is the first-order feature, and when $p \geq 2$, it is higher-order features.

their different recognition accuracy, which may not be preferable when certain classifiers are known *a priori* to perform better or worse than others. Consequently, the second disadvantage results in another undesired result that the Borda count function lacks learning ability, and will not improve its performance even when it is exposed to more and more data.

Although the purpose of class set reduction is to minimize the size of the candidate class set, if it cannot guarantee that one and only one class is contained in the result class set, it may not be applicable to some classification problems with only a small number of classes, such as numeral recognition. In general, the operation of set intersection is beneficial only when all classifiers have the correct class labels in a reasonable range of neighborhood, so that thresholds on neighborhood sizes can be selected in such a way that the candidate set can be small, while the true class is not missed. The union operation is preferred for combining a set of a highly specialized classifiers, because the strength of each classifier can be preserved, while its weakness can be compensated by other classifiers. However, it is difficult, if not impossible, to find such specialized classifiers in practice. Another disadvantage of the two methods is that only rank information is used to derive the final classification decision; no doubt, valuable measurement information is lost in the combination process. Consequently, both methods are unable to achieve the best recognition performance.

Mandler understood that it is important to utilize all measurement-level information when dealing with CME by first transforming measurement values into confidence values. But he uses a Dempster-Shafer formula to aggregate confidence values, which inherently has an independence assumption. Franke realized that all the classification outputs from every classifier should be input at the same time in order to derive the transformation matrix A. The weakness of his method is that the output of individual classifiers is used as the direct input of a polynomial classifier, without normalization. Due to the various scales and meanings of measurement values from individual classifiers, it is very difficult, or even impossible, for the polynomial classifier to derive a generalized weight matrix (A^*) which can be good for diversified cases.

Chapter 4

A New Model for Abstract-Level CME

4.1 Introduction

A thorough analysis of the reason why most abstract-level CME methods (such as voting, Bayesian, and Dempster-Shafer) require the independence assumption reveals that either each classifier is treated equally, or the probability is derived from the confusion matrix of a single classifier in isolation. To eliminate this assumption, the probability should be derived from a knowledge space which is able to record the behavior of all classifiers on each sample *concurrently* [90]. Based on this realization, we developed a novel CME method [91, 90, 92, 93, 94, 95, 96], which derives its decision from a so-called Behavior-Knowledge Space (BKS). Basically, this method operates

in two stages: (1) the knowledge-modelling stage, which extracts knowledge from the former classifiers' decisions and constructs a K-dimensional behavior-knowledge space; and (2) the decision-making stage, which combines classification decisions generated from individual classifiers for each test sample, enters into a specific cell of the constructed space, and makes a final decision by a rule which utilizes the knowledge of the cell. Excitingly, it has been shown that the BKS method possesses many advantageous properties such as (1) adaptive learning, (2) automatic threshold finding, (3) theoretical performance analysis of the combination of partial classifiers, (4) the optimal solution for abstract-level CME from the theoretical point of view, (5) better or equal performance with additional combined classifiers, and (6) no assumption that classifiers are independent of each other, so there is no degradation when dependence exists among classifiers.

Before discussing this method, one simple example is given in Section 4.2 to illustrate the basic concept. Section 4.3 makes a formal specification of a behavior knowledge space. Then Section 4.4 describes the construction of a BKS in the knowledge-modelling stage, followed by Section 4.5 which specifies the decision-making strategy in the testing process. Consecutively, three important theorems derived from the BKS method are thoroughly investigated in Section 4.6: *optimality, semi-monotonicity*, and *function dependence*. Optimality means that theoretically the BKS method is capable of producing the highest recognition rate; semi-monotonicity specifies that the

BKS method can produce a better or equal recognition performance with additional combined classifiers; and function dependence investigates whether the involvement of a new classifier will affect the recognition performance of the combination process if the decision of the new classifier is totally dependent on other classifiers. Then, Section 4.7 describes several advantageous properties of the BKS method, which account for the superiority of this method. But, two inherent problems of the BKS method may degrade considerably its practicability to practical applications. They are discussed in Section 4.8, and several solutions are proposed in this section as well. Finally, some extended discussion related to the BKS method is described in Section 4.9.

4.2 An Example to Illustrate the Basic Concept of the BKS Method

Suppose there are two persons (A and B) who are asked to answer *yes* or *no* to 400 questions. A and B do not know how many questions should be answered *yes* and how many should be answered *no*. Let there be 200 questions whose correct answers are *yes*, ("*yes*-questions"); the remaining 200 are *no*, ("*no*-questions"). Suppose, after receiving answers to all 400 questions, Table 1 is the accumulated result, of which there are four situations in total: (1) A:*yes* and B:*yes*, (2) A:*yes* and B:*no*, (3) A:*no* and B:*yes*, (4) A:*no* and B:*no*. Each situation contains a two-element pair (a, b) which

denotes two numbers of occurrence that match the answers of both A and B — a is the number of *yes*-questions and b is the number of *no*-questions. For example, when A says *yes* and B says *no*, the corresponding element is (40,10). This means that 40 of them are *yes*-questions and 10 of them are *no*-questions, *i.e.* a total of 50 questions received both *yes* from A and *no* from B. Therefore, for a new question, given that A answers *yes* and B answers *no*, if one wants to decide whether it is a *yes*-question or a *no*-question, then the best decision should be *yes* because based on past experience the probability of *yes*-questions is 80% and that of *no*-question is 20%.

Based on the above concept, it takes two steps to make a final decision. Step 1 selects one of the four situations according to the answers of A and B, and step 2 chooses which of the two kinds of questions has the highest probability as the final decision in the selected situation. Therefore, final decisions for these four situations are (1) *yes*, when A says *yes* and B says *yes*; (2) *yes*, when A says *yes* and B says *no*; (3) *yes*, when A says *no* and B says *yes*; and (4) *no*, when A says *no* and B says *no*.

Since Table 1 registers the knowledge described from the behaviors of A and B, it is called the behavior-knowledge space of A and B. The next section presents a formal discussion of a behavior knowledge space and its data representation.

A \ B	*yes*	*no*
yes	(90 , 10)	(40 , 10)
no	(60 , 20)	(10 , 160)

Table 1: Accumulated results of A and B

4.3 Behavior Knowledge Space

A BKS is a K-dimensional array of cells, where each dimension corresponds to the classification decision of one classifier and has $M+1$ possible decision values chosen from the set $\{1, 2, \cdots, M+1\}$. For classifier k, the set consisting of all its possible classification decisions is called the kth *possible decision set*. For an input pattern, if the decision of a classifier belongs to the set $\{1, \cdots, M\}$, it means that the classifier recognizes this pattern (either correctly or incorrectly); otherwise, the classifier rejects this pattern. The intersection of the classification decisions of individual classifiers occupies one cell of the BKS. In total, there are $(M+1)^K$ cells in a K-dimensional BKS; each cell accumulates the number of incoming samples for each respective class. Table 2 gives an example of a two-dimensional BKS, where cell (i, j) is the cell where $e_1(x) = i$ and $e_2(x) = j$. Each cell of a BKS contains three kinds of data: (1) the total number of incoming samples, (2) the number of incoming samples from each class, and (3) the best representative class. The learned samples distributed to one cell/class are called the *incoming samples* of that cell/class.

BKS =

$e(1) \setminus e(2)$	1	2	$\cdots$	j	$\cdots$	10	11
1	(1,1)	(1,2)	$\cdots$	(1,j)	$\cdots$	(1,10)	(1,11)
2	(2,1)	(2,2)	$\cdots$	(2,j)	$\cdots$	(2,10)	(2,11)
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
i	$\vdots$	$\vdots$	$\vdots$	(i,j)	$\vdots$	$\vdots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
10	(10,1)	(10,2)	$\cdots$	(10,j)	$\cdots$	(10,10)	(10,11)
11	(11,1)	(11,2)	$\cdots$	(11,j)	$\cdots$	(11,10)	(11,11)

Table 2: 2-D behavior-knowledge space.

Symbols related to a BKS are:

$$\begin{aligned}
\text{BKS} &= \text{a K-dimensional behavior-knowledge space,} \\
e(i) &= \text{the decision of classifier } i \quad (e(i) \in \Lambda \cup \{M+1\}), \\
\text{BKS}(e(1), \cdots, e(K)) &= \text{a cell of BKS, where classifier 1 gives its decision as } e(1), \cdots, \text{and classifier } K \text{ gives its decision as } e(K), \\
n_{e(1)\cdots e(K)}(m) &= \text{the number of incoming samples belonging to class } m \text{ in BKS}(e(1), \cdots, e(K)),
\end{aligned}$$

$$\begin{aligned}
T_{e(1)\cdots e(K)} &= \text{the total number of incoming samples in BKS}(e(1), \cdots, e(K)), \\
&= \sum_{m=1}^{M} n_{e(1)\cdots e(K)}(m), \qquad (4) \\
R_{e(1)\cdots e(K)} &= \text{the best representative class of BKS}(e(1), \cdots, e(K)), \\
&= \{j \mid n_{e(1)\cdots e(K)}(j) = max_{1 \le m \le M} n_{e(1)\cdots e(K)}(m)\}. \qquad (5)
\end{aligned}$$

For each cell, only one class should be regarded as the best representative class. If two or more classes contain the largest incoming samples, then the first of them is chosen arbitrarily as the best representative class.

4.4 The Knowledge-Modelling Stage

The objective of the knowledge-modelling stage is to construct a behavior knowledge space based on the classification decisions of the involved classifiers for all training samples.

Suppose there is a learning set L with n samples, *i.e.* $L = \{x_1, x_2, \cdots, x_n\}$, and x_i is the ith sample of L ($1 \leq i \leq n$). For one sample x_i, let $I(i)$ be its ground-truth class label and $e_1(i), \cdots, e_K(i)$ be the decisions of the K involved classifiers, respectively. Then, the following algorithm can compute the values of all parameters in a BKS:

step 1: Allocate memory to a K-dimensional BKS, and for each cell, initialize the number of the incoming sample of each class to be 0;

step 2: Set $i = 1$;

step 3: $n_{e_1(i)\cdots e_K(i)}(I(i)) := n_{e_1(i)\cdots e_K(i)}(I(i)) + 1$;

step 4: $i := i + 1$; if i is not larger than n, then go to step 3;

step 5: For each cell BKS($e(1), \cdots, e(k), \cdots, e(K)$), where $1 \leq k \leq K$ and $e(k) \in$

$\Lambda \cup \{M+1\}$, do

$$\begin{cases} T_{e(1)\cdots e(K)} = \sum_{m=1}^{M} n_{e(1)\cdots e(K)}(m), \\ R_{e(1)\cdots e(K)} = \{j \mid n_{e(1)\cdots e(K)}(j) = max_{1 \le m \le M} n_{e(1)\cdots e(K)}(m)\}. \end{cases}$$

4.5 The Decision-Making Stage

This stage shows how the BKS method derives the final classification decision from a constructed BKS. For a pattern x, let $e(1), \cdots$, and $e(K)$ represent the classification results of K classifiers, respectively. Then, according to the given information $e(1), \cdots$, and $e(K)$, we can find a special cell BKS$(e(1), \cdots, e(K))$ from the BKS. Because this cell plays an important role in deriving the final classification decision, let us first make a formal definition of it:

Definition 1: *For the current input pattern x, the cell which is the intersection of all the K classifiers' decisions is called the Focal Cell (FC).*

Therefore, the cell BKS$(e(1), \cdots, e(K))$ is the corresponding FC of a pattern x when $c_1(x) = e(1), \cdots$, and $c_K(x) = e(K)$. Then, the final decision of x will be derived from the FC by the following decision rule

$$E(x) = \begin{cases} R_{e(1)\cdots e(K)} & \text{, when } T_{e(1)\cdots e(K)} > 0 \\ & \quad \text{and } \frac{n_{e(1)\ e(K)}(R_{e(1)\ e(K)})}{T_{e(1)\ e(K)}} \ge \lambda; \\ M+1 & \text{, otherwise.} \end{cases} \tag{6}$$

where λ $(0 \le \lambda \le 1)$ is a threshold which controls the reliability of the final classification decision. In fact, in our implementation, to recognize an unlabeled pattern, the

ratio of the difference in number of incoming samples of the most and second most representative classes and the total number of incoming samples of the corresponding FC is equal to or larger than a specified threshold λ; otherwise, this pattern is rejected. However, to simplify the notation, Equation (6) will be used in the following theorem proving and property discussion.

4.6 Optimality, Semi-Monotonicity and Function Dependence of the BKS Method

If individual classifiers' behavior of training data is the same as that of testing data, then many valuable properties can be derived from this method. For example, theoretically the BKS method can produce the highest recognition rate for a given set of classifiers. This property is called the *optimality* of the BKS method. Also, the BKS method will increase its recognition performance as more classifiers are combined, which is called the *semi-monotonicity* of the BKS method. When one classifier's behavior is dependent upon others, it will not affect recognition performance, which is the *function dependence* of the BKS method. Again, it should be specifically pointed out that the following discussion is based on the assumption that the knowledge of classifier behavior derived from the constructed BKS is unbiased and genuine. In other words, these good properties are discussed from the theoretical point of view which is based on statistics derived from the data.

4.6.1 Optimality of the BKS Method

Since a few abstract-level combination modules have been developed already, it is natural to ask "Can the BKS method perform better than other combination modules?" In this sub-section, a theorem is proven which specifies that for a set of abstract-level classifiers, the BKS method is the combination function which is capable of producing the highest recognition rate. This theorem ensures the superiority of this method.

Theorem 1: *Given K abstract-level classifiers, theoretically the BKS method can produce the highest recognition rate.*

This theorem can be proven in two ways.

Proof1:

Let $BEL(i)$ be the probability that pattern x comes from class i when the K individual classifiers output their classification decisions. For abstract-level CME, when each classifier e_k gives its decision $e(k)$, then $BEL(i)$ becomes

$$BEL(i) = P(x \in C_i \,|\, e_1(x) = j_1, \cdots, e_K(x) = j_K, EN^K)$$

where $P(.)$ is the probability function and EN^K denotes a knowledge space which describes that each classifier has at most $M+1$ discrete decisions and also which stores the behavior of the K classifiers. As a matter of fact, the corresponding BKS of the K classifiers can serve EN^K. Accordingly, the situation under $\{e_1(x) = j_1, \cdots, e_K(x) = j_K$ and $EN^K\}$ is actually the same as that of the current FC, *i.e.* BKS$(j_1, \cdots, j_K)$.

Therefore, the belief function becomes

$$BEL(i) \;=\; P(x \in C_i \mid \mathrm{BKS}(j_1, \cdots, j_K)).$$

Making use of the knowledge of FC, we get

$$BEL(i) = \frac{n_{j_1,\cdots,j_K}(i)}{T_{j_1,\cdots,j_K}}.$$

According to the Bayes decision rule, a pattern should be assigned to the class having the largest *a posteriori* probability, so that the classification error is minimal. This implies that the class with the largest belief value should be chosen as the final decision. In fact, the chosen class is the class with the most numerous incoming samples. Therefore, the decision rule becomes

$$E(x) = \begin{cases} j & \text{, when } T_{e(1)\cdots e(K)} > 0 \text{ and} \\ & \quad n_{e(1)\cdots e(K)}(j) = \max_{i \in \Lambda} n_{e(1)\cdots e(K)}(i), \\ M+1 & \text{, otherwise.} \end{cases}$$

The above decision rule is identical to that of the BKS method with $\lambda = 0$ (see Equation (6)). Therefore, this concludes the proof that from the theoretical point of view, the BKS method can achieve the highest recognition rate for the combination of abstract-level classifiers. □

Proof2:

In fact, for a pattern x the decisions of K individual classifiers can construct a K-dimensional feature vector. Since the decision of each classifier is discrete and has a finite number of values, the K-dimensional feature space also has a finite number of discrete feature values. Then, the *a posteriori* probability that the input pattern belongs to class i given a feature vector $[e(1), \cdots, e(K)]$ becomes

$$P(x \in C_i \,|\, e(1), \cdots, e(K), EN^K) = \begin{cases} \frac{n_{e(1),\cdots,e(K)}(i)}{T_{e(1),\cdots,e(K)}} & \text{, if } T_{e(1),\cdots,e(K)} > 0; \\ \text{undefined} & \text{, otherwise.} \end{cases}$$

where $n_{e(1),\cdots,e(K)}(i)$ denotes the total number of occurrences that a sample belongs to class i at the feature point $(e(1), \cdots, e(1)K)$, $T_{e(1),\cdots,e(K)}$ denotes the total number of occurrences at the feature point $(e(1), \cdots, e(K))$, and EN^K denotes a discrete and finite K-dimensional feaure space which is constructed by combining the K abstract-level classifier decisions. According to the Bayes decision rule, if a pattern is assigned to the class having the largest *a posteriori* probability, then we can obtain the best recognition performance with the highest recognition rate (*i.e.* the minimun recognition error). This implies that the class with the largest number of occurrences should be chosen as the final decision. Therefore, for the minimum recognition error the

decision rule becomes

$$E(x) = \begin{cases} j & \text{, when } T_{e(1)\cdots e(K)} > 0 \text{ and} \\ & \quad n_{e(1)\cdots e(K)}(j) = \max_{i \in \Lambda} n_{e(1)\cdots e(K)}(i); \\ M+1 & \text{, otherwise.} \end{cases}$$

The above decision rule is identical to that of the BKS method with $\lambda = 0$ (see Equation (6)). Therefore, this concludes the proof that from the theoretical point of view, the BKS method can achieve the highest recognition rate for the combination of abstract-level classifiers. □

4.6.2 Semi-Monotonicity of the BKS Method

Since the BKS method will produce the best recognition result for combining multiple classifiers, then there arises a question, "Will the combination of more classifiers always produces an equal or even better decision?" In other words, will the combination of $K+1$ classifiers have an equal or even better result than that of K classifiers? Theoretically, if a method E is the globally optimal CME method for any K abstract-level classifiers and there is one more classifier involved, then the recognition performance of the $K+1$ classifiers should be equal to or even better than that of the K classifiers, because the worst case is that the $(K+1)$th classifier is just not used for the combination at all However, from a different point of view, it seems to be contradictory to our intuition. In human society, if one person is not qualified for a specific task or has a bad reputation with regard to his/her working attitude,

it is proper not to include this person in a team, because he/she may impede the progress of the team. Fortunately, this impediment will not happen from the theoretical point of view when adopting the BKS method. The following gives a formal proof to confirm the validity of the *semi-monotonicity* property.

Theorem 2: *Using the BKS method, theoretically the combination of* $K+1$ *classifiers will produce an equal or even better result than that of* K *classifiers.*

Proof: This can be proven by induction.

Step 1: When $K = 0$ (which means that there is no classifier), then each class gets the same possibility $\frac{1}{M}$ of having the input pattern. When one needs to decide which class this pattern belongs to, no matter which class is chosen, the average recognition rate will be $\frac{1}{M}$ and the average substitution rate is $\frac{M-1}{M}$. If one classifier is included, then there exist $M+1$ cells in the corresponding BKS, *i.e.* BKS(1),$\cdots$, BKS(i),$\cdots$, and BKS($M+1$). For any cell BKS(i), $1 \leq i \leq M+1$, suppose its best representative class is R_i, then

$$R_i = \{j \,|\, n_i(j) = max_{1 \leq m \leq M} n_i(m)\}.$$

Obviously,

$$n_i(R_i) \geq n_i(m), \qquad \forall m \in \Lambda.$$

Therefore,

$$M * n_i(R_i) \geq \sum_{1 \leq m \leq M} n_i(m).$$

Then,

$$\frac{n_i(R_i)}{\sum_{1\le m\le M} n_i(m)} \ge \frac{1}{M}.$$

The above equation specifies that the recognition rate for each cell BKS(i), $1 \le i \le M+1$, is equal to or larger than $\frac{1}{M}$. Accordingly, the final recognition rate produced by this classifier is also equal to or larger than $\frac{1}{M}$. This shows that "The statement is true when $K = 0$".

Step 2: Suppose the statement is true when $K = n$ (n is a positive integer). Then each cell in the corresponding BKS is in the form BKS($e(1), \cdots, e(K)$). When another classifier is added, then each cell of the corresponding BKS becomes BKS($e(1)$, $\cdots$, $e(K)$,$e(K+1)$) . In fact, each cell BKS($e(1), \cdots, e(K)$) is further decomposed into $M+1$ cells, *i.e.* BKS($e(1), \cdots, e(K), 1$), $\cdots$, BKS($e(1), \cdots, e(K), i$), $\cdots$, and BKS($e(1), \cdots, e(K), M+1$). We call cell BKS($e(1), \cdots, e(K)$) the parent of cells BKS($e(1), \cdots, e(K), i$), where $i \in \{1, \cdots, M+1\}$.

Let $R_{e(1)\cdots e(K)}$ stand for the best representative class of cell BKS($e(1), \cdots, e(K)$) and $R_{e(1)\cdots e(K)i}$ stand for the best representative class of cell BKS($e(1), \cdots, e(K), i$); then

$$R_{e(1)\cdots e(K)i} = \{j \mid n_{e(1)\cdots e(K)i}(j) = max_{1\le m\le M} n_{e(1)\cdots e(K)i}(m)\}.$$

Obviously,

$$n_{e(1)\cdots e(K)i}(R_{e(1)\cdots e(K)i}) \ge n_{e(1)\cdots e(K)i}(m), \qquad \forall m \in \Lambda. \tag{7}$$

Because $R_{e(1)\cdots e(K)} \in \Lambda$, therefore

$$n_{e(1)\cdots e(K)i}(R_{e(1)\cdots e(K)i}) \geq n_{e(1)\cdots e(K)t}(R_{e(1)\cdots e(K)}).$$

The above equation means that for cell BKS$(e(1), \cdots, e(K), i)$, the number of incoming samples for its best representative class is always equal to or larger than that for any class, which surely includes the best representative class of BKS$(e(1), \cdots, e(K))$. Therefore,

$$\sum_{1 \leq t \leq M+1} n_{e(1)\cdots e(K)i}(R_{e(1)\cdots e(K)i}) \geq \sum_{1 \leq t \leq M+1} n_{e(1)\cdots e(K)t}(R_{e(1)\cdots e(K)}).$$

The right part of the above equation is indeed equal to $n_{e(1)\cdots e(K)}(R_{e(1)\cdots e(K)})$, so it becomes

$$\sum_{1 \leq t \leq M+1} n_{e(1)\cdots e(K)i}(R_{e(1)\cdots e(K)i}) \geq n_{e(1)\cdots e(K)}(R_{e(1)\cdots e(K)}).$$

Dividing the above equation by $T_{e(1)\cdots e(K)}$, then we obtain

$$\frac{\sum_{1 \leq t \leq M+1} n_{e(1)\cdots e(K)i}(R_{e(1)\cdots e(K)i})}{T_{e(1)\cdots e(K)}} \geq \frac{n_{e(1)\cdots e(K)}(R_{e(1)\cdots e(K)})}{T_{e(1)\cdots e(K)}}.$$

Since the summation of the incoming samples for cells BKS$(e(1), \cdots, e(K), 1)$, $\cdots$, and BKS$(e(1), \cdots, e(K), M+1)$ is the same as the number of the incoming samples $T_{e(1)\cdots e(K)}$ of cell BKS$(e(1), \cdots, e(K))$, i.e. $T_{e(1)\cdots e(K)} = \sum_{i=1}^{M+1} T_{e(1)\cdots e(K)i}$, then we get

$$\frac{\sum_{1 \leq t \leq M+1} n_{e(1)\cdots e(K)i}(R_{e(1)\cdots e(K)i})}{\sum_{i=1}^{M+1} T_{e(1)\cdots e(K)i}} \geq \frac{n_{e(1)\cdots e(K)}(R_{e(1)\cdots e(K)})}{T_{e(1)\cdots e(K)}}. \quad (8)$$

Equation (8) shows that the average recognition rate of all children cells is equal to or greater than that of their parent cell. Since the recognition rate of each cell is equal

to or smaller than the average recognition rate of its children cells, it becomes clear that the recognition rate of the $n+1$ classifiers is equal to or larger than that of the n classifiers. This shows that "the above statement is true when $K = n$ (n is any positive integer)".

Step 3: Therefore, by induction, we can state that "the recognition rate of $K+1$ classifiers is equal to or larger than that of K classifiers for any K". This concludes the proof. □.

We may ask in what situations the performance of $K+1$ classifiers will equal that of K classifiers. Obviously, the answer comes from Equation (7) when only the equal relation is kept, *i.e.*

$$n_{e(1)\cdots e(K)i}(R_{e(1)\cdots e(K)i}) = n_{e(1)\cdots e(K)i}(R_{e(1)\cdots e(K)}) \qquad \forall i \in \{1, \cdots, M+1\}. \quad (9)$$

Interestingly enough, two cases have been shown to satisfy Equation (9):

(1) the $(K+1)$th classifier has a totally random behavior; and

(2) the behavior of the $(K+1)$th classifier depends on those of the other K classifiers, *i.e.* $e(K+1) = f(e(1), \cdots, e(K))$, where f represents a function.

For example, suppose there are 90 incoming samples in BKS$(e(1), \cdots, e(K)$, 80 samples belonging to class 4 and 10 to class 5. Obviously, R$(e(1), \cdots, e(K))$ is 4. In case 1, there will be 9 samples in each BKS$(e(1), \cdots, e(K), i)$ for all $i \in \{1, \cdots, M\}$; among them, 8 and 1 samples belong to classes 4 and 5, respectively. Therefore,

Equation (9) is true. In case 2, only one possible value will be generated from the $(K+1)$th classifier when $e(1), \cdots,$ and $e(K)$ are given. Suppose this value is I; then all incoming samples in cell BKS$(e(1), \cdots, e(K))$ will become the incoming samples of cell BKS$(e(1), \cdots, e(K), I)$, and no samples will enter cell BKS$(e(1), \cdots, e(K), i)$ for all $i \neq I$ and $i \in \{1, \cdots, M+1\}$. Undoubtedly, Equation (9) is true in this case as well.

4.6.3 Function Dependence of the BKS Method

The above discussion reveals an interesting observation: if the classification decisions of a classifier are totally dependent upon those of other classifiers, then from the BKS point of view, this classifier offers no improvement to the recognition accuracy of combination at all. This phenomenon will be described in Theorem 3.

Theorem 3: *Given $K+1$ abstract-level classifiers, if the behavior of the $(K+1)$th classifier is totally dependent upon the results of the other K classifiers, then adding the $(K+1)$th classifier will not improve the recognition accuracy.*

Proof:

Suppose classifier $(K+1)$ behaves totally dependent on the other K classifiers, this means that the decisions of classifier $(K+1)$ can be described exactly by the decisions of the other K classifiers through a function f. Therefore, whenever the decisions of the other K classifiers are given, the decisions of classifier $(K+1)$ will

be generated definitely. This dependence can be expressed as

$$e(K+1) = f(e(1), \cdots, e(K)).$$

It is easy to verify that all incoming samples of BKS$(e(1), \cdots, e(K))$ will also be the incoming samples of BKS$(e(1), \cdots, e(K), e(K+1))$, and there are no incoming samples which enter BKS$(e(1), \cdots, e(K), i)$ if $i \neq e(K+1)$. Therefore, using the decision rule of Equation (6), obviously the recognition performance of the $K+1$ classifiers is the same as that of the K classifiers. □.

4.7 Other Advantageous Properties

This section describes four other advantageous properties of the BKS method, namely *adaptive learning, automatic threshold finding, theoretical performance analysis*, and *no assumption that classifiers are independent of each other.*

4.7.1 Adaptive learning

Learning is an important function of any recognition system. This function can gradually improve the system's performance by continuously accumulating more input data. Adaptive learning means that the algorithm can learn and adapt its knowledge to the real behavior of the given (particular) application. The learning ability can be implemented as follows:

Suppose I is the ground-truth class label of the input pattern x, and $e(1), \cdots, e(K)$

represent the classification decisions of the K classifiers to x; then the learning algorithm of the BKS method will be:

1. $n_{e(1)\cdots e(K)}(I) := n_{e(1)\cdots e(K)}(I) + 1$,

2. $T_{e(1)\cdots e(K)} := T_{e(1)\cdots e(K)} + 1$,

3. $R_{e(1)\cdots e(K)} = \{j \mid n_{e(1)\cdots e(K)}(j) = max_{1 \le m \le M} n_{e(1)\cdots e(K)}(m)\}$.

4.7.2 Automatic Threshold Finding

Since the performance of an OCR system usually is not perfect (*i.e.* 100% recognition and 0% substitution rates), different performances are required for different applications. For example, the substitution rate should be very close to 0% in monetary applications, but slight errors are tolerable for address reading, such as one error per 1000 pieces of mail. Therefore, it is important that a system can automatically adapt itself to its required performance. The goal here is to find a threshold P_t for Equation (6) which assures that the system will perform at its required level. Let $C(P_t)$ be defined as an error function which is the weighted sum of three values, each of which is the square of the distance between required and derived rates, that is

$$\begin{aligned} C(P_t) &= \text{error function} \\ &= \alpha * (\text{D\_C} - \mathcal{C})^2 + \beta * (\text{D\_S} - \mathcal{S})^2 + \gamma * (\text{D\_R} - \mathcal{R})^2 \end{aligned}$$

where α is the weight for the recognition rate,

β is the weight for the substitution rate,

γ is the weight for the rejection rate,

$\mathcal{C}$ is the required recognition rate,

$\mathcal{S}$ is the required substitution rate,

$\mathcal{R}$ is the required rejection rate,

D_C is the derived recognition rate,

D_S is the derived substitution rate, and

D_R is the derived rejection rate.

In general, $\alpha + \beta + \gamma = 1.0$. The values of α, β, γ, $\mathcal{C}$, $\mathcal{S}$ and $\mathcal{R}$ should be supplied for a specific application, and considered as system requirements. Of course, the sum of $\mathcal{C}$, $\mathcal{S}$ and $\mathcal{R}$ is equal to 1.0. The values of D_C, D_S and D_R can be derived from the BKS by Equation (6) with threshold P_t; their sum is 1.0 as well. To compute D_C, D_S and D_R, four new variables are defined below:

$$\begin{aligned} N &= \text{the total number of incoming samples in the whole BKS,} \\ &= \sum_{\epsilon(1)=1}^{M} \cdots \sum_{\epsilon(K)=1}^{M} T_{\epsilon(1)\cdots\epsilon(K)}; \end{aligned}$$

$$\begin{aligned} D_{\epsilon(1)\cdots\epsilon(K)} &= \text{the proportion that cell BKS}(\epsilon(1),\cdots,\epsilon(K)) \text{ to the whole BKS,} \\ &= \frac{T_{\epsilon(1)\cdots\epsilon(K)}}{N}; \end{aligned}$$

$P_{c(1)\ c(K)}$ = the probability of the best representative class in BKS($c(1), \cdots, c(K)$),

$$= \frac{n_{c(1)\ c(K)}(R_{c(1)\ c(K)})}{T_{c(1)\ c(K)}};$$

and,

$f(P_{c(1)\cdots c(K)}, P_t)$ = the acceptance index function,

$$= \begin{cases} 1 & \text{, when } P_{c(1)\ c(K)} \geq P_t, \\ 0 & \text{, otherwise.} \end{cases}$$

For a cell BKS($c(1), \cdots, c(K)$), there are two exclusive situations between $P_{c(1)\ c(K)}$ and P_t:

(1) $P_{c(1)\cdots c(K)} < P_t$, all the samples in this cell are rejected; therefore, its influence on both D_C and D_S is 0, and on D_R is $D_{c(1)\ c(K)}$. In this situation, $f(P_{c(1)\ c(K)}, P_t) = 0$.

(2) $P_{c(1)\cdots c(K)} \geq P_t$, all samples in this cell are accepted; $T_{c(1)\ c(K)} * P_{c(1)\ c(K)}$ samples are correctly recognized and $T_{c(1)\cdots c(K)} * (1 - P_{c(1)\cdots c(K)})$ samples are misrecognized. Therefore, its influence on D_C is $D_{c(1)\cdots c(K)} * P_{c(1)\ c(K)}$, on D_S is $D_{c(1)\cdots c(K)} * (1 - P_{c(1)\cdots c(K)})$, and on D_R is 0. In this situation, $f(P_{c(1)\ c(K)}, P_t) = 1$.

In fact, the two situations can be combined into one more general form: for a cell BKS($c(1)\cdots c(K)$), its influence on D_C is $D_{c(1)\cdots c(K)} * P_{c(1)\ c(K)} * f(P_{c(1)\ c(K)}, P_t)$, on D_S is $D_{c(1)\cdots c(K)} * (1 - P_{c(1)\cdots c(K)}) * f(P_{c(1)\cdots c(K)}, P_t)$, and on D_R is $D_{c(1)\ c(K)} * (1 - f(P_{c(1)\cdots c(K)}, P_t))$. Therefore, summing up the influences of each cell in the BKS

gives

$$\text{D\_C} = \sum_{c(1)=1}^{M+1} \cdots \sum_{c(K)=1}^{M+1} D_{c(1)\ \cdots\ c(K)} * P_{c(1)\ \cdots\ c(K)} * f(P_{c(1)\ \cdots\ c(K)}, P_t),$$

$$\text{D\_S} = \sum_{c(1)=1}^{M+1} \cdots \sum_{c(K)=1}^{M+1} D_{c(1)\ \cdots\ c(K)} * (1 - P_{c(1)\ \cdots\ c(K)}) * f(P_{c(1)\ \cdots\ c(K)}, P_t),$$

$$\text{D\_R} = \sum_{c(1)=1}^{M+1} \cdots \sum_{c(K)=1}^{M+1} D_{c(1)\ \cdots\ c(K)} * (1 - f(P_{c(1)\ \cdots\ c(K)}, P_t)).$$

After substituting the derived values of D_C, D_S and D_R into Equation (10), the error function $C'(P_t)$ becomes

$$\begin{aligned} C'(P_t) = \quad & \alpha * [\sum_{c(1)=1}^{M+1} \cdots \sum_{c(K)=1}^{M+1} D_{c(1)\ \cdots\ c(K)} * P_{c(1)\ \cdots\ c(K)} * f(P_{c(1)\ \cdots\ c(K)}, P_t) - \mathcal{C}]^2 \\ & + \beta * [\sum_{c(1)=1}^{M+1} \cdots \sum_{c(K)=1}^{M+1} D_{c(1)\ \cdots\ c(K)} * (1 - P_{c(1)\ \cdots\ c(K)}) * f(P_{c(1)\cdots c(K)}, P_t) - \mathcal{S}]^2 \\ & + \gamma * [\sum_{c(1)=1}^{M+1} \cdots \sum_{c(K)=1}^{M+1} D_{c(1)\ \cdots\ c(K)} * (1 - f(P_{c(1)\ \cdots\ c(K)}, P_t)) - \mathcal{R}]^2 \end{aligned} \tag{10}$$

where P_t is the only variable which is unknown. Obviously, the smaller the value of $C'(P_t)$ is, the closer the derived performance will be to the desired one. Therefore, the best P_t (denoted by P^*) is the one which minimizes the value of the error function, *i.e.* $C'(P^*) = \min_{0 \le P_t \le 1} C'(P_t)$. There are several optimization algorithms which are capable of finding P^*, *e.g.* simulated annealing [97] and random optimization [98]. In short, whenever $\alpha, \beta, \gamma, \mathcal{C}, \mathcal{S}$, and $\mathcal{R}$ are known, the best threshold P^* can be found automatically by using Equation (10).

Although, Equation (10) can derive the best threshold for the required performance, P^* may not be the real best threshold from the viewpoint of an OCR system. This can be clearly illustrated by an example. Suppose $\alpha = \beta = \gamma = \frac{1}{3}$, $\mathcal{C} = 98\%$,

$\mathcal{S} = 1\%$ and $\mathcal{R} = 1\%$; then the following two cases will produce the same value of error functions:

case 1: D_C = 99%, D_S = 0% and D_R = 1%;

case 2: D_C = 97%, D_S = 2% and D_R = 1%.

The error value of the two cases is $C(P_t) = \frac{1}{3}(1+1) = \frac{2}{3}$. Since the higher recognition rate and the lower substitution rate are preferred in an OCR system, the result of case 1 is indeed much better than that of case 2. This indicates that a situation where either D_C is larger than C or D_S is smaller than S should not only have no increment, but also should reduce the error function. Also, the rejection rate can be derived when both the recognition and substitution rates are known, *i.e.*,

$$\text{rejection rate} = 100\% - \text{recognition rate} - \text{substitution rate}.$$

Based on the consideration of the above two points, a probable modified error function becomes

$$C(P_t) = \alpha * g_1(\mathcal{C}, \text{D\_C}) + \beta * g_2(\mathcal{S}, \text{D\_S}) \tag{11}$$

where

$$g_1(\mathcal{C}, \text{D\_C}) = \begin{cases} (\text{D\_C} - \mathcal{C})^2 & , \text{ when } \mathcal{C} \geq \text{D\_C}, \\ -(\text{D\_C} - \mathcal{C})^2 & , \text{ otherwise;} \end{cases}$$

and

$$g_2(\mathcal{S}, \text{D\_S}) = \begin{cases} (\text{D\_S} - \mathcal{S})^2 & \text{, when } \mathcal{S} \leq \text{D\_S}, \\ -(\text{D\_S} - \mathcal{S})^2 & \text{, otherwise.} \end{cases}$$

Sometimes one does not know exactly the appropriate values for α and β. However, C is usually required to be much larger than $\mathcal{S}$. Many applications require high reliability and cannot endure substitution rates over 0.1%, otherwise it becomes too risky to use such OCR systems. For these applications, β should be very large. In general, the lower $\mathcal{S}$ is, the larger β should be. Accordingly, an intuitive suggestion for α and β is to set them to the reciprocal of the square of their individual corresponding desired rates, that is $\alpha = \frac{1}{C^2}$ and $\beta = \frac{1}{\mathcal{S}^2}$.

4.7.3 Theoretical Performance Analysis of the Combination of a Partial Set of Classifiers

Theoretically, combining more classifiers will probably produce a better recognition performance as stated in Theorem 2. However, more powerful hardware is required to enhance the speed as well. In reality, the performance and the cost of the supporting hardware should be considered at the same time, *i.e.* the trade-off between the number of classifiers and the cost of the supporting hardware. Therefore, how to compute the performance of a variable number of classifiers and how to find the subset of classifiers which can perform best are very important topics.

Suppose n classifiers have been selected from the original K classifiers ($n \leq K$). Then, the main operation to solve this problem is to constitute a new n-dimensional

BKS of the n selected classifiers from the K-dimensional BKS, where the corresponding performance of the n classifiers can be derived. In the following discussion, let (1) $q = \{i_1, i_2, \cdots, i_n\}$ represent the index-sequence set of the n classifiers, where i_1 denotes the i_1th classifier, $\cdots$, i_n denotes the i_nth classifier, and $q \subseteq \{1, \cdots, K\}$; (2) $\overline{q}$ represent the complementary set of the index-sequence set q over $\{1, 2, \cdots, K\}$, i.e. $\overline{q} = \{j_1, \cdots, j_{K-n}\} = \{j \mid j \notin q, j \in \{1, \cdots, K\}\}$; (3) BKS_K and BKS_n denote a K-dimensional and an n-dimensional behavior-knowledge space respectively; and (4) h be a funtion which has a K-dimensional input domain and an n-dimensional output range. For example, suppose the first and the third classifiers have been chosen among four classifiers ($K = 4$), then $q = \{1, 3\}$ and $\overline{q} = \{2, 4\}$. Then the mapping can be formally expressed as

$$h : \mathrm{BKS}_K \longrightarrow \mathrm{BKS}_n$$

where h forms a cell $\mathrm{BKS}_n(e(i_1), \cdots, e(i_n))$ by merging (denoted by operator $\bigcup$) all the cells with the same decision values of the corresponding dimensions $\{i_1, \cdots, i_n\}$ of BKS_K. That is

$$\mathrm{BKS}_n(e(i_1), \cdots, e(i_n)) = \bigcup_{(j \in \overline{q}) \text{ and } (e(j) \in \{1, \cdots, M+1\})} \mathrm{BKS}_K(e(1), \cdots, e(j), \cdots, e(K))$$

where $\bigcup$ produces $n'_{e(i_1) \cdots e(i_n)}(m)$ and $T'_{e(i_1) \cdots e(i_n)}$ as

$$n'_{e(i_1) \cdots e(i_n)}(m) = \sum_{e(j_1)=1}^{M+1} \cdots \sum_{e(j_k)=1}^{M+1} \cdots \sum_{e(j_{K-n})=1}^{M+1} n_{e(1) \cdots e(K)}(m)$$

and

$$T'_{e(i_1)\cdots e(i_n)} = \sum_{e(j_1)=1}^{M+1} \cdots \sum_{e(j_k)=1}^{M+1} \cdots \sum_{e(j_{K-n})=1}^{M+1} T_{e(1)\cdots e(K)}$$

where $1 \le m \le M$, $j_k \in \bar{q}$ and $1 \le k \le K-n$.

Accordingly, the best representative class for $\mathrm{BKS}_n(e(i_1), \cdots, e(i_n))$ is

$$R'_{e(i_1)\cdots e(i_n)} = \{j \mid n'_{e(i_1)\cdots e(i_n)}(j) = max_{1 \le m \le M} n'_{e(i_1)\cdots e(i_n)}(m)\}.$$

Among K classifiers, there are a total of $C(K,n) = \frac{K!}{n!(K-r)!}$ different combinations of n classifiers. Let $q^r = \{i_1^r, \cdots, i_n^r\}$ correspond to the rth combination; then the performance index of q^r is defined as

$$PER(q^r) = \frac{\sum_{e(i_1^r)=1}^{M+1} \cdots \sum_{e(i_n^r)=1}^{M+1} n'_{e(i_1^r)\cdots e(i_n^r)}(R'_{e(i_1^r)\cdots e(i_n^r)})}{\sum_{e(i_1^r)=1}^{M+1} \cdots \sum_{e(i_n^r)=1}^{M+1} T'_{e(i_1^r)\cdots e(i_n^r)}}. \tag{12}$$

In fact, $PER(q^r)$ is the recognition rate of the combination of classifiers $i_1^r, \cdots, i_n^r$ with no rejection. Let q^s correspond to the sth combination which produces the highest performance index; then

$$PER(q^s) = \max_{1 \le r \le C(K,n)} PER(q^r). \tag{13}$$

Obviously, the n classifiers belonging to q^s will have the best performance among any n classifiers, and should be selected to derive the final decision.

In practice, if combining 3 classifiers has already satisfied the performance requirement for the current application, then it is not necessary to add any other classifier to improve the system performance. Based on this realization, the following algorithm

can deduce the best combination with the minimal number of classifiers to satisfy the required performance, by using the properties discussed in sub-sections 4.7.2 and 4.7.3 together. For a more general purpose, let η denote an error-tolerance threshold, indicating the maximum acceptable error between derived and required performances. Given a set of classifiers, the following algorithm has the ability to select the minimum number of classifiers which are capable of deriving the final recognition decision with an average error less than η.

step 1: Input the values of α, β, γ, $\mathcal{C}$, $\mathcal{S}$, $\mathcal{R}$ and η;

step 2: Set $n = 1$;

step 3: Compute the performance indices of all combinations of n out of K classifiers by Equation (12);

step 4: Compute the best combination (q^s) by Equation (13);

step 5: Compute the error function $C(P_t)$ of set q^s and its corresponding threshold value P^* by Equation (11);

step 6: IF $C(P^*)$ is smaller than η, THEN the solution has been found and it **stops**;

step 7: IF n is equal to K (the total number of classifiers), THEN it **stops** with a FAILURE output, ELSE $n = n + 1$;

step 8: Go to step 3.

When the algorithm stops with a FAILURE signal, it means that the requested system performance is unable to achieve even when all the K classifiers are combined; otherwise, n indicates the number of classifiers which should be combined and q^s indicates the index-sequence set of the corresponding classifiers.

4.7.4 No Requirement of Classifier Independence

Methods such as voting, Bayesian, and Dempster-Shafer assume that the behavior of individual classifiers is independent of each other. In practice, their performances may degrade considerably if this assumption is not satisfied. However, the BKS method will not suffer from this drawback, because it does not assume that classifiers are independent of each other. This argument will become clearer through the following discussion.

Suppose there are $K+1$ classifiers, and the behavior of the $(K+1)$th classifier totally depends on those of the other K classifiers. From Theorem 3, we know that all incoming samples of BKS$(e(1),\cdots,e(K))$ also enter BKS$(e(1),\cdots, e(K), f(e(1),\cdots,e(K)))$, where f is a function mapping $e(1),\cdots,$ and $e(K)$ to $e(K+1)$, that is $e(K+1) = f(e(1),\cdots,e(K))$. Therefore, the recognition performance of combining the $K+1$ classifiers is the same as that of combining the other K classifiers.

Although, from the theoretical point of view, the dependence between classifiers does not degrade the recognition accuracy, it does require extra memory to produce many *redundant* cells (a cell with no incoming samples is called a redundant cell).

Broadly speaking, the number of redundant cells is proportional to the degree of dependence among classifiers. This means that the higher dependence exhibited among classifiers, the larger the number of redundant cells will be.

4.8 Problems and Proposed Solutions

The above has described the BKS method and its advantageous properties. So far, this method seems to be perfect, without drawbacks. In fact, there are two issues which could seriously affect its practicality: (1) whether the BKS method can perform effectively when there are not enough learning samples, and (2) according to the nature of the K-dimensional BKS, its required memory is exponential, and will become intolerably high when K is large. In this section, useful solutions are proposed to address these two issues, so that the practicality of this method can be ensured. These solutions also show the flexibility in managing the knowledge of a BKS.

4.8.1 Not enough learning samples

The efficiency of the KBS method depends on the representativeness of a BKS to the real classifiers' behavior. Therefore, if the learning samples are not numerous or representative enough to make the BKS fully represent the classifiers' behavior, then the KBS method cannot maintain the optimality of Theorem 1. Differing behavior between training and testing samples is indeed the most serious problem before the BKS

method can be reliably applied to real applications. However, this problem can be solved by incorporating the BKS method with another abstract-level CME method A (*e.g.* voting, Bayesian, or Dempster-Shafer). Heuristically, the method producing the best recognition performance among all other abstract-level CME methods should be chosen as method A. Accordingly, whenever there is useful information (*i.e.* $T_{e(1)\ \cdots\ e(K)} > 0$) in the corresponding focal cell (*FC*), then the final decision will be computed by the BKS method (Equation (6)); otherwise, the decision should be computed by the chosen method A. The final decision can be made as follows:

$$E(x) = \begin{cases} R_{e(1)\cdots e(K)} & , \frac{n_{e(1)\ \cdots\ e(K)}(R_{e(1)\ \cdots\ e(K)})}{T_{e(1)\ \cdots\ e(K)}} \geq \alpha \text{ and } T_{e(1)\cdots e(K)} \geq 0; \\ E(x) \text{ of } A & , T_{e(1)\cdots e(K)} = 0; \\ 0 & , \text{otherwise.} \end{cases}$$

In fact, if too few samples (*e.g.* less than 3) are accumulated in a cell, then the behavior representativeness of this cell will not be sufficient enough to make a reliable decision. To reduce the risk of making decisions in such a situation, a number threshold β should be set in order to make more reliable decisions. Therefore, the decision rule is modified to

$$E(x) = \begin{cases} R_{e(1)\cdots e(K)} & , \frac{n_{e(1)\ \cdots\ e(K)}(R_{e(1)\ \cdots\ e(K)})}{T_{e(1)\ \cdots\ e(K)}} \geq \alpha \text{ and } T_{e(1)\cdots e(K)} \geq \beta; \\ E(x) \text{ of } A & , \max_A > \lambda \text{ and } T_{e(1)\cdots e(K)} < \beta; \\ 0 & , \text{otherwise.} \end{cases}$$

where β is a number threshold which determines whether the current *FC* should make a definite decision or not, λ is a reliability threshold for method A, and $\max_A$ stands

for the largest belief value to a certain class derived from method A.

4.3.2 Exponential memory requirement

Exponential memory requirement is a crucial constraint of the BKS method. In the case of numeral recognition with four classifiers, the total number of cells is $11^4 = 14641$. Since each cell contains the number of incoming samples for each digit, the BKS needs at least $14641 * 10$ integers to store the information. The size of the BKS will increase rapidly if more classifiers are to be combined. Therefore, it is very important to find solutions which can reduce the exponential memory requirement. Three solutions have been proposed and are described below.

Solution 1: Dynamic class allocation

Originally, each cell allocates a certain amount of memory to each class to count the individual incoming samples. However, each class does not always have incoming samples. Thus, it should be more reasonable to allocate memory to a class in a cell only when necessary. This means that instead of allocating memory to M classes in each cell, if some classes do not have any incoming sample, no memory will be allocated to them.

Solution 2: Dynamic cell allocation

A K-classifier BKS is a K-dimensional space. Usually, it is an extremely sparse space when K is large, *i.e.* many cells do not have any incoming sample. With the same concept of Solution 1, it should only allocate memory to the necessary cells.

This means that instead of allocating memory to all cells, if some cells do not have any incoming samples, no memory will be allocated to them.

However, when the cells are dynamically allocated, it is necessary to have an efficient mechanism to reorganize these allocated but scattered cells. Obviously, the mechanism must have two basic abilities: (1) to insert or delete a cell, and (2) to search for any cell. In Solution 3, we will see that it is necessary to maintain the additional comprehension of being able to delete a cell. For the purpose of searching cells, a cell should be represented by a unique index value. As a result, the index value for a cell BKS($c(1), \cdots, c(K)$) can be expressed as

$$\begin{aligned} \text{index}(\text{BKS}(c(1)\cdots c(K)) &= \textstyle\sum_{k=1}^{K} c(k) * (M+1)^{k-1}, \\ &= c(1) + c(2) * (M+1) + \cdots + c(K) * (M+1)^{K-1}. \end{aligned}$$

With the indices of cells, several tree construction algorithms (such as AVL trees [99] or B^+ trees [100]) are appropriate for organizing the scattered cells, if they satisfy the following two requirements:

1. The algorithm constructs a balanced tree which takes about the same amount of time to search any leaf index;

2. It can easily insert or delete a leaf index.

Solution 3: Condensed editing technique

This solution is motivated by the concept "*keep only the necessary knowledge in BKS*". This means that whenever a decision made by the BKS method is the

same as that made by another CME method which requires less memory, then the corresponding cell of the BKS will be deleted from the constructed tree, because the same decision can be derived from the other CME method. Intuitively, the method chosen in Section 4.8.1 could also be chosen here. Suppose method A is the chosen method and $E(x \mid e_1(x), \cdots, e_K(x), EN^K)$ stands for a combination function when K classifiers give their decisions as $e_1(x), \cdots$, and $e_K(x)$ respectively; then the constructed index tree can be pruned by the following rule:

IF $E(x \mid e_1(x), \cdots, e_K(x), EN^K)$ of the BKS method

$= E(x \mid e_1(x), \cdots, e_K(x), EN^K)$ of method A,

THEN delete the node with index $(e(1), \cdots, e(K))$ from the constructed tree.

After performing the condensed editing technique, the decision rule supported by both the BKS method and method A becomes

IF index$(e(1), \cdots, e(K))$ exists in the constructed tree,

$$\text{THEN} \quad E(x) = \begin{cases} R_{e(1)\cdots e(K)} & , \frac{n_{e(1)\ \ e(K)}(R_{e(1)\ \ e(K)})}{T_{e(1)\ \ e(K)}} \geq \alpha; \\ \\ M+1 & , \text{otherwise;} \end{cases}$$

$$\text{ELSE} \quad E(x) = \begin{cases} j & , \text{if } (Bel(j) = max_{i \in \Lambda} Bel(i)) \geq \lambda; \\ M+1 & , \text{otherwise;} \end{cases}$$

where α and λ are thresholds with $0 \leq \alpha, \lambda \leq 1$, and $Bel(i)$ is the belief value

computed by method A

4.9 Further Discussion

4.9.1 Will the BKS Method Benefit from Consecutive CME?

As described before, consecutive CME is based on the hypothesis that since CME may achieve higher classification accuracy than individual classifiers, by further combining the results of different CME approaches even higher classification accuracy may be achieved. Heuristically, this hypothesis seems to be quite reasonable. However, from the theoretical point of view, because the BKS method has already achieved the globally optimal classification accuracy for a given set of classifiers, it is intrinsically impossible for another CME method to obtain a higher classification accuracy than the BKS method for the same set of classifiers. Obviously, there is a conflict between human heuristic and theoretical understanding; this conflict may be clarified through the following discussion.

Suppose there are K classifiers $e_1, \cdots, e_K$, and n abstract-level CME methods whose combination modules are $A_1, \cdots$, and A_n, respectively. Let O_i be the output of A_i, $i \in \{1, \cdots, n\}$. Then,

$$O_i = A_i(e_1, \cdots, e_K).$$

This means that O_i is functionally dependent upon $e_1, \cdots$, and e_K. Therefore, when the decisions of the K classifiers are given, then A_i can only produce one unique

result. From Theorem 3, it is easy to verify that the recognition accuracy of combining $e_1, \cdots, e_K$ is the same as that of combining both $e_1, \cdots, e_K$ and $A_1, \cdots, A_n$. Accordingly, the combination result of the K classifiers will equal that by using both the K classifiers and the n CME modules.

4.9.2 More Training Samples and Different Initialization

The success of the BKS method depends on the representativeness of the behavior of training samples with respect to that of testing samples. One commonly-used approach to increase the representativeness is to enlarge the number of training samples. Naturally, a leave-one-out estimation [13] might be adopted to test recognition performance: *all samples but one are learned and the unlearned one is tested.* The same operation is repeated until every sample in the data set has been left out and tested. In general, the leave-one-out estimation can produce the most unbiased recognition performance for a given set of training samples. However, for the BKS method, when applying the leave-one-out estimation, a low substitution rate may be difficult to achieve. This can be explained clearly by the following example: suppose one cell BKS(4,9) has 4 incoming samples; among them, three belong to digit "4", and one (name it B) to digit "9". Obviously, the leave-one-out estimation always makes one error, no matter what value threshold α may be. This is because when B is the one left out, the BKS method will decide with total confidence that each sample in this cell is digit "4".

This phenomenon can be overcome by two schemes. The first is simply to collect more training samples: obviously, if there are more samples to make those irregular cases repeatable, then the substitution rate should be more sensitive to α and more easily reduced. For example, suppose the number of incoming samples of this cell is doubled, so that 6 incoming samples belong to digit "4", and 2 incoming samples to digit "9". If the threshold α is higher than $\frac{6}{7}$, then all 8 samples of this cell will not be recognized but rejected. However, this approach is not pertinent to some applications, because it may be too difficult or expensive to collect more samples for them. The second scheme is to initialize the number of incoming samples of each class in each cell to 1, instead of 0. Using the same example above (*i.e.* 3 samples belong to digit "4" and 1 sample to digit "9"), if the sample belonging to digit "9" is left out, then $n_{4,9}(4) = 3 + 1 = 4$ and $n_{4,9}(9) = 1$. Accordingly, if the threshold α is larger than $\frac{4}{5}$, then this sample will be rejected. Experiments have shown the effectiveness of these two schemes.

4.9.3 Extension of Ambiguous Classification Decisions

For an unlabeled pattern x, suppose there are two classifiers, and classifier 1 (c_1) cannot make a single choice but decides that x may be from either class A or class B, and classifier 2 (c_2) gives its classification decision to class C decisively. In other words, $c_1 = \{A \vee B\}$ and $c_2 = C$. Therefore, the belief that x belongs to class i will

be expressed as

$$BEL(i) = P(x \in C_i \mid e_1 = \{A \vee B\}\ e_2 = C, EN^2).$$

If there is an element corresponding to the decision $\{A \vee B\}$ in the possible decision set of the first classifier, then the above probability can be computed from a two-dimensional BKS directly. Therefore, the decision rule of Equation (6) can be applied to derive the final decision as well, because theoretically ambiguous decisions can be combined by using the BKS method without difficulty. However, in practice, there are three aspects which make the above scheme hard to implement. First, for a given classifier, we cannot guarantee that all ambiguous decisions of unseen samples will appear in training samples. In other words, some ambiguous decisions may not appear in training samples, but only in testing samples. So it is impossible to encode all ambiguous decisions into the corresponding possible decision set. Secondly, to include all ambiguous decisions in a BKS, much more computer memory is required to maintain a BKS, which is already a serious problem to the BKS method. Thirdly, many training samples are required in order to make several repetitions of the rare ambiguous classification decisions. Accordingly, another scheme is proposed to serve this end: in the knowledge-modelling stage, for each classifier only a subset A of all ambiguous classification decisions is considered in its possible decision set. Among the training samples, each element of subset A appears more frequently than the other ambiguous decisions not in the subset. In other words, only the most frequent

ambiguous decisions are encoded in the corresponding possible decision set. Then, by using training samples and the possible decision sets of all classifiers, a corresponding BKS is then constructed. In the decision-making stage, if the decisions made by all classifiers can be found in their corresponding possible decision set, then Equation (6) can serve to derive the final classification decision directly; otherwise a different decision-making approach will be applied. A possible approach for decision making is proposed below.

Before specifying this approach, the above example is used to illustrate the basic concept of this approach: there are two classifiers, e_1 and e_2, and for an input pattern x, $e_1 = \{A \vee B\}$ and $e_2 = C$. Accordingly, $BEL(i)$ becomes

$$\begin{aligned} BEL(i) &= P(x \in C_i \mid e_1 = \{A \vee B\}, e_2 = C, EN^2) \\ &= P(x \in C_i \mid e_1 = \{A \vee B\} \wedge e_2 = C \wedge EN^2) \\ &= \frac{P(x \in C_i \wedge e_1 = \{A \vee B\} \wedge e_2 = C \mid EN^2)}{P(e_1 = \{A \vee B\} \wedge e_1 = C \mid EN^2)}. \end{aligned} \tag{14}$$

Let D and N represent the numerator and denominator of the above equation respectively. To tackle the expression $e_1 = \{A \vee B\}$, we decompose it into the expression $e_1 = A \vee e_1 = B$, since any pattern can only genuinely belong to one class. With this decomposition, then

$$\begin{aligned} D &= P(x \in C_i \wedge e_1 = \{A \vee B\} \wedge e_2 = C \mid EN^2) \\ &= P(x \in C_i \wedge (e_1 = A \vee e_1 = B) \wedge e_2 = C \mid EN^2) \\ &= P((x \in C_i \wedge e_1 = A \wedge e_2 = C) \vee (x \in C_i \wedge e_1 = B \wedge e_2 = C) \mid EN^2) \end{aligned}$$

and

$$\begin{aligned} N &= P(e_1 = \{A \vee B\} \wedge e_2 = C \mid EN^2) \\ &= P((e_1 = A \wedge e_2 = C) \vee (e_1 = B \wedge e_2 = C) \mid EN^2) \end{aligned}$$

In probability theory,

$$P(X \vee Y) = P(X) + P(Y) - P(X \wedge Y)$$

where X and Y are two hypotheses. When X and Y are almost exclusive, then the value of $P(X \wedge Y)$ is very small, and can be omitted. If this condition is satisfied, then

$$P(X \vee Y) = P(X) + P(Y).$$

In character recognition, only few cases produce results with both $e_1 = A \wedge e_2 = C$ and $e_1 = B \wedge e_2 = C$ at the same time. Therefore, it seems to be safe to say that these two results are approximately exclusive. So,

$$D = P(x \in C_i \wedge e_1 = A \wedge e_2 = C \mid EN^2) + P(x \in C_i \wedge e_1 = A \wedge e_2 = C \mid EN^2).$$

Similarly,

$$N = P(e_1 = A \wedge e_2 = C \mid EN^2) + P(e_1 = B \wedge e_2 = C \mid EN^2). \qquad (15)$$

Since $P(X \wedge Y) = P(X \mid Y)P(Y)$, the first item of D becomes

$$\begin{aligned} &P(x \in C_i \wedge e_1 = A \wedge e_2 = C \mid EN^2) \\ &= P(x \in C_i \mid e_1 = A \wedge e_2 = C \wedge EN^2) * P(e_1 = A \wedge e_2 = C \mid EN^2), \end{aligned}$$

and the second item of D becomes

$$P(x \in C_i \wedge e_1 = B \wedge e_2 = C \mid EN^2)$$

$$= P(x \in C_i \mid e_1 = B \wedge e_2 = C \wedge EN^2) * P(e_1 = B \wedge e_2 = C \mid EN^2).$$

Therefore, D becomes

$$\begin{aligned} D &= P(x \in C_i \mid e_1 = A \wedge e_2 = C \wedge EN^2) * P(e_1 = A \wedge e_2 = C \mid EN^2) \\ &\quad + P(x \in C_i \mid e_1 = B \wedge e_2 = C \wedge EN^2) * P(e_1 = B \wedge e_2 = C \mid EN^2). \end{aligned} \quad (16)$$

Using Equations (15) and (16), Equation (14) becomes

$$\begin{aligned} BEL(i) &= \frac{P(x \in C_i \mid e_1 = A \wedge e_2 = C \wedge EN^2) * P(e_1 = A \wedge e_2 = C \mid EN^2)}{P(e_1 = A \wedge e_2 = C \mid EN^2) + P(e_1 = B \wedge e_2 = C \mid EN^2)} + \\ &\quad \frac{P(x \in C_i \mid e_1 = B \wedge e_2 = C \wedge EN^2) * P(e_1 = B \wedge e_2 = C \mid EN^2)}{P(e_1 = A \wedge e_2 = C \mid EN^2) + P(e_1 = B \wedge e_2 = C \mid EN^2)} \\ &= w_1 * P(x \in C_i \mid e_1 = A \wedge e_2 = C \wedge EN^2) + \\ &\quad w_2 * P(x \in C_i \mid e_1 = B \wedge e_2 = C \wedge EN^2) \end{aligned} \quad (17)$$

where

$$w_1 = \frac{P(e_1 = A \wedge e_2 = C \mid EN^2)}{P(e_1 = A \wedge e_2 = C \mid EN^2) + P(e_1 = B \wedge e_2 = C \mid EN^2)},$$

$$w_2 = \frac{P(e_1 = B \wedge e_2 = C \mid EN^2)}{P(e_1 = A \wedge e_2 = C \mid EN^2) + P(e_1 = B \wedge e_2 = C \mid EN^2)}.$$

it is worthwhile to note that $w_1 + w_2 = 1$. Indeed, w_1 and w_2 serve the weight parameters which indicate the possibilities that the input pattern comes from units BKS(A, C) and BKS(B, C), respectively.

For another example, $e_1 = \{A \vee B\}$ and $e_2 = \{C \vee D\}$. By using the above derivation procedure, it is easy to derive the following equation.

$$\begin{aligned} BEL(i) \quad = \quad & w_1 * P(x \in C_i \mid e_1 = A \wedge e_2 = C \wedge EN^2) \\ & + w_2 * P(x \in C_i \mid e_1 = B \wedge e_2 = C \wedge EN^2) \\ & + w_3 * P(x \in C_i \mid e_1 = A \wedge e_2 = D \wedge EN^2) \\ & + w_4 * P(x \in C_i \mid e_1 = B \wedge e_2 = D \wedge EN^2) \end{aligned}$$

where

$$w_1 = \frac{P(e_1=A\wedge e_2=C \mid EN^2)}{P(e_1=A\wedge e_2=C \mid EN^2)+P(e_1=B\wedge e_2=C \mid EN^2)+P(e_1=A\wedge e_2=D \mid EN^2)+P(e_1=B\wedge e_2=D \mid EN^2)},$$

$$w_2 = \frac{P(e_1=B\wedge e_2=C \mid EN^2)}{P(e_1=A\wedge e_2=C \mid EN^2)+P(e_1=B\wedge e_2=C \mid EN^2)+P(e_1=A\wedge e_2=D \mid EN^2)+P(e_1=B\wedge e_2=D \mid EN^2)},$$

$$w_3 = \frac{P(e_1=A\wedge e_2=D \mid EN^2)}{P(e_1=A\wedge e_2=C \mid EN^2)+P(e_1=B\wedge e_2=C \mid EN^2)+P(e_1=A\wedge e_2=D \mid EN^2)+P(e_1=B\wedge e_2=D \mid EN^2)},$$

$$w_4 = \frac{P(e_1=B\wedge e_2=D \mid EN^2)}{P(e_1=A\wedge e_2=C \mid EN^2)+P(e_1=B\wedge e_2=C \mid EN^2)+P(e_1=A\wedge e_2=D \mid EN^2)+P(e_1=B\wedge e_2=D \mid EN^2)}.$$

Therefore, when there are K classifiers and each classifier k produces classification decisions $e_k = \{j_{k1}, \cdots, j_{kI_k}\}$, where classes $j_{k1}, \cdots, j_{kI_k}$ are those classes that e_k highly recommends and I_k is the total number of class labels supported by e_k, then the aggregated belief $BEL(i)$ become

$$BEL(i) = \sum_{i1=1}^{I_1} \cdots \sum_{iK=1}^{I_K} w_{j_1^{i1} \cdots j_K^{iK}} P(x \in C_i \mid e_1 = j_1^{i1}, \cdots, e_K = j_K^{iK}, EN^K),$$

where

$$w_{j_1^{i1} \cdots j_K^{iK}} = \frac{P(e_1 = j_1^{i1}, \cdots, e_K = j_K^{iK} \mid EN^K)}{\sum_{i1=1}^{I_1} \cdots \sum_{iK=1}^{I_K} w_{j_1^{i1} \cdots j_K^{iK}} P(e_1 = j_1^{i1}, \cdots, e_K = j_K^{iK} \mid EN^K)}.$$

After the belief values for all classes are computed, then various decision rules can be applied to derive the final decision.

4.9.4 An Advantageous Tool For Further Refinement

Shridhar *et al.* [62] argued that sequential CME, in general, does not perform as well as parallel CME. However, the concept of sequential CME can be applied to improve the performance of a recognition system after parallel CME. For an input pattern x, after the recognition of individual classifiers and the parallel CME process, if x is highly likely to pertain to more than one class, then it is not appropriate to make a decisive classification decision in this situation. A most common solution is simply to reject this pattern; but another possible solution is to design a corresponding refining algorithm A, which further analyzes the distinctive aspects among the images of different classes appearing in this cell. As a result, algorithm A will produce the final classification decision. However, In general this job is very expensive to carry out. Fortunately, with the help of a BKS, algorithm A can probably be implemented much more easily and systematically. Basically, each cell requires one algorithm to distinguish its confusion classes. There are three reasons why a BKS can benefit from the development of algorithm A: (1) A cell can not only count how many incoming samples, but also which samples enter this cell, therefore with an image displaying program, these samples can be visualized so that the difference among them can be observed explicitly; (2) For a specific cell, usually only one, two or three classes have non-zero numbers of incoming samples; therefore, the complexity of algorithm A is intrinsically reduced; and (3) In general, the samples of a class in the same cell

tend to have specific shapes; this phenomenon again has the potential to simplify the development of algorithm A. An experiment has been performed to justify the above claims. With three classifiers and 22,021 numeral patterns, a three-dimensional BKS is constructed. It contains $11^3 = 1331$ cells, and Table 3 lists the the number of cells with respect to the number of different classes which have non-zero incoming samples appearing in a cell. When $c_1 = 3$, $c_2 = 2$, and $c_3 = 2$, there are 100 samples in cell BKS(3,2,2). Figure 5 shows all their corresponding images. Obviously, it is much easier to design an algorithm to distinguish two classes than an algorithm for all classes. However, although the cells of a BKS enable us to develop algorithm A faster, it may be impractical or too expensive to design a corresponding algorithm for each cell. Broadly speaking, there are two kinds of cells which do not really affect the overall recognition performance. The first kind includes cells which contain a very small number of incoming samples, such as only 1, 2 or 3 incoming samples. The second is the cells which have many incoming samples, but most of which belong to a single class. Therefore, one alternative is to design algorithms only for those cells having many incoming samples, and most of which do not belong to a single class. For example, with the requirement that $T_{c(1)c(2)c(3)} > 20$ and $\frac{n_{c(1)c(2)c(3)}(R_{c(1)c(2)c(3)})}{T_{c(1)c(2)c(3)}} < 0.6$, then 14 cells need to design such further discrimination algorithms. In total, there are 792 samples in these cells. Among them, by the BKS method with no rejection, there are 277 incorrectly recognized samples. Therefore with 14 proper algorithms,

classes	1	2	3	4	5	6	7	8	9
n	868	263	137	42	6	1	0	0	0

Table 3: The distribution of the number of classes *versus* the number of cells per class with three classifiers and 22,024 numeral samples, where *classes* denotes the number of classes having non-zero incoming samples, and n the number of cells having non-zero incoming samples.

these 277 samples may be recovered correctly; that is 1.26% of the testing samples, a significant improvement in the overall recognition rate.

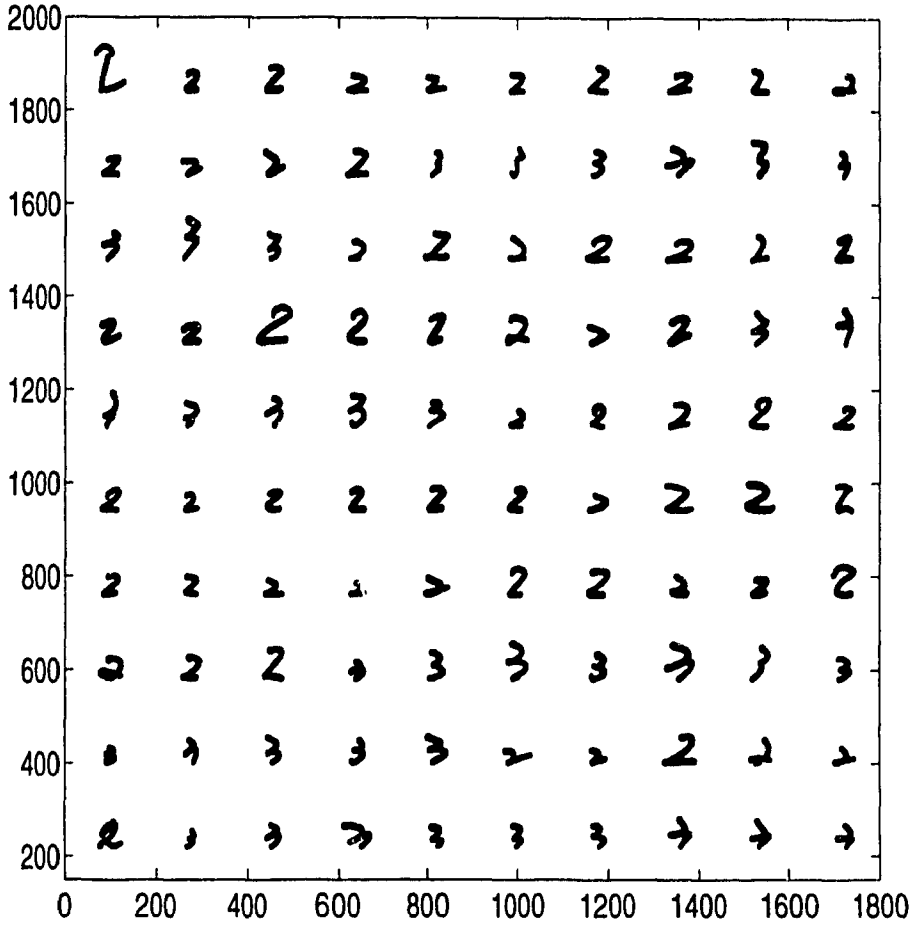


Figure 5: Display of the incoming samples in cell BKS(3,2,2).

Chapter 5

Methods for Measurement-Level CME

5.1 Introduction

According to two different views of the roles that individual classifiers play in a multi-classifier system, there are different approaches to the research of measurement-level CME. The first is to view individual classifiers as *classifiers*, and the second is to view them as *feature extractors*. Accordingly, the research of the first view focuses on how to transform measurement values into reliability-like values, and then how to aggregate the transformed values together [101]. Based on the second point of view, CME becomes a generic pattern recognition problem; therefore, the basic research goal is to develop classification methods which can produce high recognition

performance from the measurement values supported by individual classifiers.

In this chapter, three methods will be proposed; two are designed from the first point of view, and the third is from the second point of view. Section 5.2 introduces the two methods based on the first viewpoint, the Linear and Bayesian Confidence Aggregation methods (LCA and BCA). Section 5.3 introduces a neural-network based approach with a multi-layer perceptron to combine measurement values of all classifiers, which is designed from the second point of view; the important topics of this method are the data transformation function, network architecture, and net learning algorithm. Finally, in Section 5.4, three strategies are proposed to improve the performance of multi-layer perceptrons, in both recognition speed and accuracy.

5.2 Two Confidence Aggregation Methods: LCA and BCA

Two methods to combine measurement-level classifiers are proposed in this section, Linear and Bayesian Confidence Aggregation methods. Both consist of three steps: first, measurement values offered by individual classifiers are transformed into *probability*-based confidence values between [0-1]; second, a confidence aggregation function then computes the overall confidence value with respect to each class; and third, based on the accumulated overall confidence values, the final classification decision is derived by a decision rule. Broadly speaking, LCA aggregates individual

confidence values in a linear form, and BCA in a Bayesian-based form. Both methods use similar data transformation functions and decision rules, described in Section 5.2.1 and Section 5.2.2, respectively. Finally, a brief discussion about these two methods is given in Section 5.2.3, which points out their intrinsic weaknesses.

5.2.1 The Linear Confidence Aggregation Method

Let $BEL(i)$ denote the aggregated score for class i, which indicates the likelihood that a pattern comes from class i, and E be an aggregating function. Then

$$\begin{aligned} BEL(i) &= \text{the } \textit{aggregated belief} \text{ for } x \text{ being in } C_i, \\ &= E(i, e_1(x), \cdots, e_K(x)), \\ &= E(i, \{m_1^1, \cdots, m_1^M\}, \cdots, \{m_K^1, \cdots, m_K^M\}). \end{aligned} \tag{18}$$

Suppose m_k^i only influences $BEL(i)$. In other words, a measurement value to class i will not influence other classes except class i. Then Equation (18) becomes

$$BEL(i) = E(i, m_1^i, \cdots, m_k^i, \cdots, m_K^i). \tag{19}$$

The commonly-used and simplest model of function E is to perform a weighted and linear summation, such as

$$BEL(i) = w_1 * m_1^i + \cdots + w_k * m_k^i + \cdots + w_K * m_K^i \tag{20}$$

where w_k is the weight of classifier k ($1 \leq k \leq K$). Usually, these weights are adjusted only in the design phase, and remain fixed during operation. The weights are chosen

by the system designers according to their experience or, sometimes, by statistical measurements. Due to the potential nonlinearity among data, a constant weight in fact cannot serve its role well. Instead, it should be a function in which the value is computed with respect to the measurement value m_k^i. Based on this understanding, a modified aggregation function becomes

$$BEL(i) = t_1(m_1^i) + \cdots + t_k(m_k^i) + \cdots + t_K(m_K^i) \tag{21}$$

where t_k denotes the function of classifier k which transforms a measurement value into a *reliability*-like confidence value. Since this approach sums up all confidence values linearly, it is called the Linear Confidence Accumulation method (LCA).

In fact, functions $t_1, \cdots, t_K$ of Equation (21) could be expressed by a general function CF. Obviously, CF contains three parameters: the index of classifier k, label of class i , and measurement value m_k^i. Therefore, Equation (21) becomes

$$\begin{aligned} BEL(i) &= CF(1, i, m_1^i) + \cdots + CF(K, i, m_K^i) \\ &= \sum_{k=1}^{K} CF(k, i, m_k^i). \end{aligned} \tag{22}$$

The purpose of $CF(k, i, m_k^i)$ is to compute the reliability or probability that a pattern x belongs to class i when classifier k gives measurement value m_k^i to this class. Heuristically, $CF(k, i, m_k^i)$ can be expressed and computed by a conditional probability as

$$CF(k, i, m_k^i) = P(x \in C_i \mid m_k^i, EN_k^i). \tag{23}$$

The rest of this sub-section describes the procedure for computing the value of $CF(k,i,m_k^i)$ from training samples.

Suppose there are N training samples, each x of which contains both $I(x)$ and $\{m_k^i \mid \forall i,k\ (1 \leq i \leq M \text{ and } 1 \leq k \leq K)\}$, where $I(x)$ is the ground-truth class label of x, and m_k^i is the measurement value with respect to class i produced by classifier k. Accordingly, $CF(k,i,m_k^i)$ can be expressed by a conditional probability as

$$\begin{aligned} CF(k,i,m_k^i) &= P(x \in C_i \mid m_k^i, EN_k^1) \\ &= P(i = I(x) \mid m_k^i, EN_k^1). \end{aligned} \tag{24}$$

Let P_k be the probability function of classifier k; Equation (24) becomes

$$CF(k,i,m_k^i) = P_k(i = I(x) \mid m_k^i). \tag{25}$$

Using the property of a conditional probability, $CF(k,i,m_k^i)$ can be further derived as

$$\begin{aligned} CF(k,i,m_k^i) &= \frac{P_k(i = I(x), m_k^i)}{P_k(m_k^i)} \\ &= \frac{P_k(i = I(x), m_k^i)}{P_k(i = I(x), m_k^i) + P_k(i \neq I(x), m_k^i)} \\ &= \frac{R_k^i(m_k^i)}{R_k^i(m_k^i) + S_k^i(m_k^i)} \\ &= \frac{R_k^i(m_k^i)}{T_k^i(m_k^i)} \end{aligned} \tag{26}$$

where $R_k^i(m_k^i) + S_k^i(m_k^i) = T_k^i(m_k^i)$. $R_k^i(m_k^i), S_k^i(m_k^i)$ and $T_k^i(m_k^i)$ are respectively the total number of samples of which i is the ground-truth class label $I(x)$, the total

number of samples of which i is not the ground-truth class label, and the total number of samples of which classifier k supports class i when given a measurement value m_k^i.

Since m_k^i is a real number, Equation (26) cannot be computed directly in a digital computer. Therefore, m_k^i will first be quantized into one of the predefined levels $\{1, \cdots, L_k\}$, where L_k is the total number of discrete levels for classifier k. Let r_k be the quantization function of classifier k which quantizes m_k^i into level s; then $R_k^i(s), S_k^i(s)$ and $T_k^i(s)$ can be computed by ALGORITHM 1. Here, $R_k^i(s)$ is the total number of samples where i is the ground-truth class label $I(x)$, when classifier k supports class i with a measurement value in level s. Similarly, $S_k^i(s)$ and $T_k^i(s)$ have their new definitions with respect to the quantized level s. Accordingly, $CF(k, i, s)$ becomes

$$CF(k, i, s) = \frac{R_k^i(s)}{T_k^i(s)}. \tag{27}$$

{ initialization }

FOR $k := 1$ to K do

FOR $i := 1$ to M do

FOR $s := 1$ to L_k do

$R_k^i(s) := S_k^i(s) := 0;$

{ calculate the values of $R_k^i(s), S_k^i(s)$ and $T_k^i(s)$}

FOR $k := 1$ to K do

BEGIN

FOR $n := 1$ to N do

FOR $i := 1$ to M do

BEGIN

$s := r_k(m_k^i(x_n))$; {quantization of $m_k^i(x_n)$}

IF $i = I(x_n)$

THEN $R_k^i(s) := R_k^i(s) + 1;$

ELSE $S_k^i(s) := S_k^i(s) + 1;$

END

FOR $i := 1$ to M do

FOR $s := 1$ to L_k do

$T_k^i(s) := R_k^i(s) + S_k^i(s);$

END

ALGORITHM 1: the computation of $R_k^i(s), S_k^i(s)$ and $T_k^i(s)$ from an N-size learning set.

If there are a sufficient number of learning samples (*e.g.* 5000 samples / digit), then the derived $R_k^i(s)$, $S_k^i(s)$ and $T_k^i(s)$ may be representative and meaningful; otherwise these values are unreliable for practical use. To overcome this problem, three solutions have been proposed: (1) to enlarge the learning set, (2) to use a small number (instead of a large number) of levels, and (3) to derive confidence values not dependent upon class label, but dependent only upon each classifier. The first solution, in general, gets the best result, but it also requires the heaviest workload. As a matter of fact, for some applications it is impossible to collect more training samples. The second and third solutions can be implemented easily, but they will make $CF(k, i, m_k^i)$ less sensitive to either measurement value m_k^i or individual class label i. When the third solution is adopted, $CF(k, i, m_k^i)$ is computed as

$$CF(k, i, s) = \frac{\sum_{i=1}^{M} R_k^i(s)}{\sum_{i=1}^{M} T_k^i(s)}. \tag{28}$$

Obviously, with the above Equation (28), $CF(k, i, s)$ is insensitive to class label i. Accordingly, this insensitivity causes a loss in discrimination ability. As a result, the second solution becomes the most reasonable one to apply. But, with a small number of discrete levels, the effectiveness of the measurement values is intrinsically reduced as well. This phenomenon is seen in Figure 6, which shows a stair-like confidence distribution where each quantization level corresponds to a range of measurement values. Mathematical models have been found useful to represent the confidence distributions so that this drawback can be overcome. For example, the confidence

distribution of Figure 6 becomes a smooth curve when a sigmoidal function is used to approximate the original distribution shown in Figure 7. In general, a sigmoidal function G_k which serves as the mathematical model of classifier k can be expressed as

$$G_k(y) = 1 - \frac{1}{1 + e^{\frac{-(y-a_k)}{b_k}}}. \tag{29}$$

In Equation (29), the parameter a_k serves as a threshold or a bias. The effect of a positive a_k is to shift the function to the right along the horizontal axis, and the effect of b_k is to modify the shape of the sigmoidal. A low value of b_k tends to give the sigmoidal a sharp rise, whereas a high value results in a more gently varying function. With a least-mean-square-error estimation to minimize $\mathrm{E}\{[G(m_k^i) - CF(k,i,m_k^i)]^2\}$, a_k and b_k can be derived from training samples.

In summary, the transformation from measurement values to confidence values can be carried out in five steps using the following procedure:

1. Divide the range of the measurement values supported by each classifier into a set of broad discrete intervals;

2. Compute $R_k^i(s), S_k^i(s)$ and $T_k^i(s)$ by ALGORITHM 1 from the training data set;

3. Compute $CF(k,i,s)$ by Equation (27);

4. Find a mathematical function G_k (such as a sigmodial model) which can best fit the distribution $CF(k,i,s)$, and adapt its parameters by minimizing $\mathrm{E}\{(G_k(m_k^i)-$

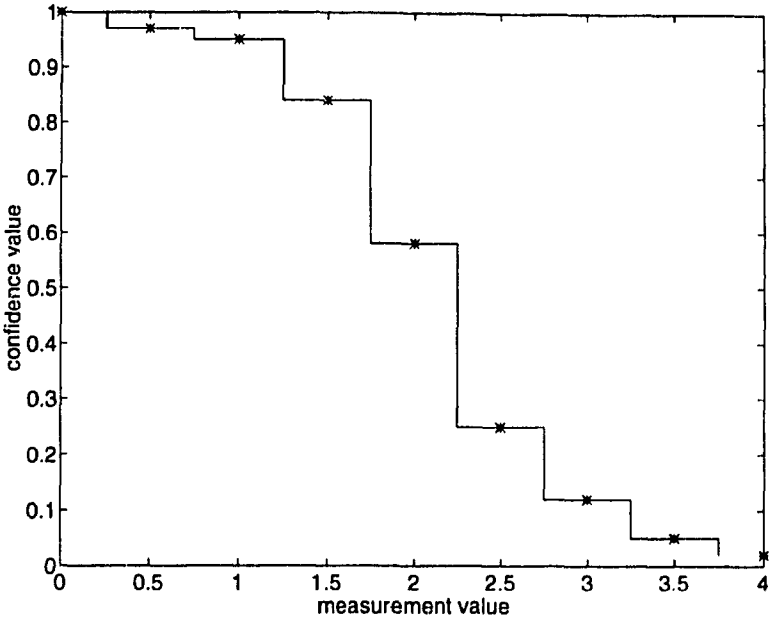


Figure 6: An example of stair-like confidence distribution, where symbols "$*$" indicate the middle positions of stairs.

$CF(k,i,m_k^1)\}$, where E denotes the expectation value;

5. For each input, compute the confidence value $CF(k,i,m_k^1)$ by function $G_k(m_k^1)$ (*i.e.* $CF(k,i,m_k^1) = G(m_k^1)$).

5.2.2 The Bayesian Confidence Aggregation Method

In general, a linear summation of individual confidence values cannot reflect the true overall beliefs if the corresponding applications require nonlinear aggregation of confidence values. Unfortunately, in practice, the relation of individual confidence

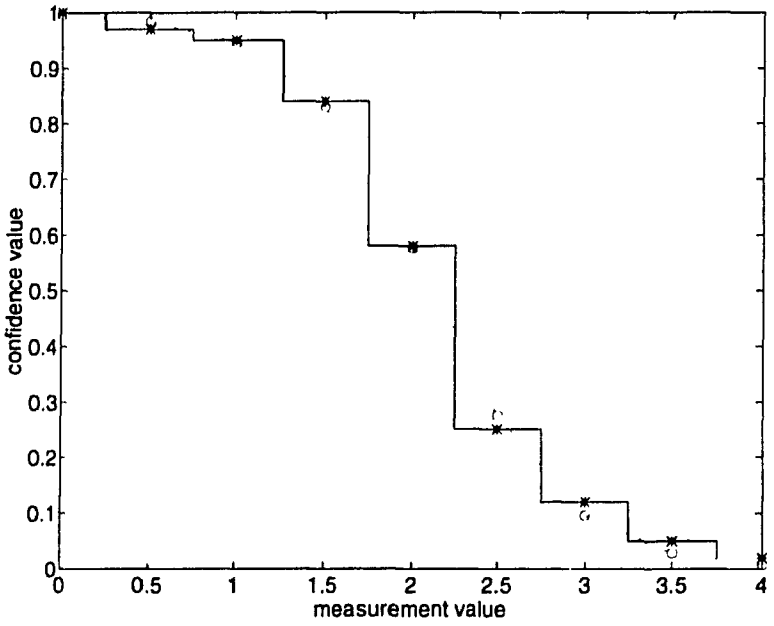


Figure 7: The simulated confidence distribution of Figure 6 calculated by a sigmoidal function, where symbols "o" are the corresponding middle positions.

values of most applications are not linear. Therefore, a new function to generate the overall beliefs nonlinearly is implemented and discussed in this section. Since this function is derived by using both the Bayesian formula and the probability-based confidence values, it is called the Bayesian Confidence Aggregation Method (BCA). So far, BCA can only take into consideration the measurement value of the first-choice class supported by individual classifiers.

Let $e_k = (j_k, m_k^{j_k})$ indicate that the first choice of classifier k to an input x is class j_k with a measurement $m_k^{j_k}$. Then $BEL(i)$ can be expressed by a conditional probability as

$$BEL(i) = P(x \in C_i \mid e_1 = (j_1, m_1^{j_1}), \cdots, e_K = (j_K, m_K^{j_K}), EN^K). \tag{30}$$

By the Bayesian Formula, Equation (30) becomes

$$BEL(i) = \frac{P(e_1 = (j_1, m_1^{j_1}), \cdots, e_K = (j_K, m_K^{j_K}) \mid x \in C_i, EN^K) * P(x \in C_i \mid EN^K)}{P(e_1 = (j_1, m_1^{j_1}), \cdots, e_K = (j_K, m_K^{j_K}) \mid EN^K)}.$$

Suppose classifiers $e_1, \cdots, e_K$ are independent of each other; then we have

$$BEL(i) = \frac{\Pi_{k=1}^{K} P(e_k = (j_k, m_k^{j_k}) \mid x \in C_i, EN_k^1)}{\Pi_{k=1}^{K} P(e_k = (j_k, m_k^{j_k}) \mid EN_k^1)} P(x \in C_i \mid EN^k) \tag{31}$$

where EN_k^1 denotes the classification environment generated from classifier k alone. Let e_k^c denote the class label supported by classifier k, and e_k^m denote the measurement value supported by classifier k. Suppose e_k^c and e_k^m are also independent of each other, *i.e.* the recognized class label has no correlation to its corresponding measurement

value; then

$$P(e_k = (j_k, m_k^{j_k}) \mid x \in C_i, EN_k^1) = P(e_k^c = j_k \mid x \in C_i, EN_k^1) * P(e_k^m = m_k^{j_r} \mid x \in C_i, EN_k^1) \tag{32}$$

and

$$P(e_k = (j_k, m_k^{j_k}) \mid EN_k^1) = P(e_k^c = j_k \mid EN_k^1) * P(e_k^m = m_k^{j_k} \mid EN_k^1). \tag{33}$$

By using the Bayesian formula, we obtain

$$P(e_k^c = j_k \mid x \in C_i, EN_k^1) = \frac{P(x \in C_i \mid e_k^c = j_k, EN_k^1) * P(e_k^c = j_k \mid EN_k^1)}{P(x \in C_i \mid EN_k^1)} \tag{34}$$

and

$$P(e_k^m = m_k^{j_k} \mid x \in C_i, EN_k^1) = \frac{P(x \in C_i \mid e_k^m = m_k^{j_k}, EN_k^1) * P(e_k^m = m_k^{j_k} \mid EN_k^1)}{P(x \in C_i \mid EN_k^1)}. \tag{35}$$

Applying Equations (32), (33), (34) and (35) to Equation (31), it becomes

$$BEL(i) = \frac{\Pi_{k=1}^K P(x \in C_i \mid e_k^c = j_k, EN_k^1) * P(x \in C_i \mid e_k^m = m_k^{j_k}, EN_k^1)}{\Pi_{k=1}^K P^2(x \in C_i \mid EN_k^1)} * P(x \in C_i \mid EN^K). \tag{36}$$

In Equation (36), $P(x \in C_i \mid e_k^c = j_k, EN_k^1)$ can be computed from the confusion matrix of classifier k; $P(x \in C_i \mid EN^K)$ is equal to $P(x \in C_i \mid EN_k^1)$ for all $k \in \{1, \cdots, K\}$, and its value is the ratio of the number of samples in class i to the total number of samples in the learning data set. $P(x \in C_i \mid e_k^m = m_k^{j_k}, EN_k^1)$ can be computed by a procedure similar to that used to calculate $P(x \in C_i \mid m_k^i, EN_k^1)$. However, there is a slight difference between them, because $P(x \in C_i \mid m_k^i, EN_k^1)$

represents the probability that a pattern x belongs to class i when classifier k gives measurement value m_k^i to this very class, but $P(x \in C_i \mid c_k^m = m_k^{j_k}, EN_k^1)$ represents the probability that a pattern x belongs to class i when classifier k gives measurement value m_k^i to class j_k. In other words, for $P(x \in C_i \mid c_k^m = m_k^{j_k}, EN_k^1)$, a measurement value to class j_k will have a certain influence on the probability that pattern x belongs to class j even when $j \neq j_k$, and it can be computed similarly, as described in ALGORITHM 1.

Accordingly, $BEL(i)$ of Equation (36) is computable. Usually, each class in the learning set has the same number of samples; then the denominator of Equation (36) is immaterial, and can be deleted. In general, the summation of $BEL(i)$ for all class labels is set to one, thus we obtain

$$BEL(i) = \eta \prod_{k=1}^{K} P(x \in C_i \mid c_k^c = j_k, EN_k^1) * P(x \in C_i \mid c_k^m = m_k^{j_k}, EN_k^1) \qquad (37)$$

where η is the normalization coefficient which makes $\sum_{i=1}^{M} BEL(i) = 1$.

5.2.3 Intrinsic Weakness of LCA and BCA

It is easy to verify that a few assumptions are required during the derivation of LCA and BCA. Among them, some are required by both methods, and some particularly by only one of them. These assumptions include:

- Classifiers behave independently of each other;
- A measurement value for class i has influence only on the belief of the very class i;

- Confidence values are aggregated together linearly;
- Measurement values are independent of class labels.

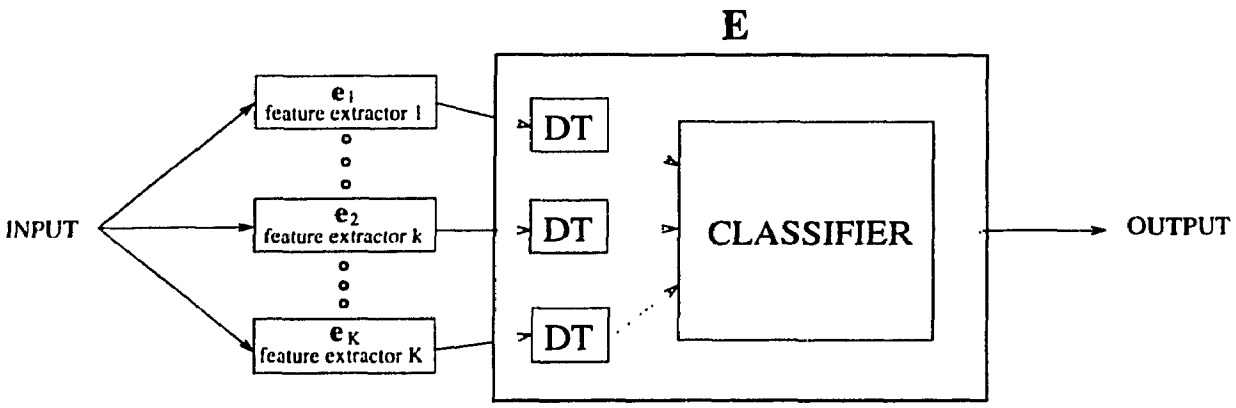
Furthermore, BCA can be applied only when each classifier supports the measurement value of its first-choice class. These requirements and constraints considerably reduce the effectiveness of the two confidence aggregation methods. The reason that both methods require the listed assumptions is that all operations to simplify the original equations are designed heuristically; as a result, they contain many built-in constraints or assumptions. In order to avoid these assumptions, in a combination function, the inter-relation between measurement values and class labels should be derived automatically from the measurement data of training samples, instead of making intuitive assumptions from human heuristics. This consideration has motivated us to propose the third method for measurement-level CME.

5.3 A Novel Approach Based on Neural-Network

In a multi-expert recognition system, individual classifiers function not only as *character classifiers* but also as *feature extractors* [102, 57]. When they function as feature extractors, their outputs become features for later classification. From this point of view, measurement-level CME becomes a pattern recognition problem, and the combination function E indeed turns out to be a generic classification function. Figure 8 shows the block diagram of this new consideration. With this understanding,

not surprisingly, neural networks can surely be applied to serve the function of E, whose input consists of the transformed measurement values and output consists of M indication values, each of which indicates the likelihood that the input belongs to certain class. Intrinsically, neural networks are suitable for measurement-level CME, because they contain four well-known valuable characteristics: (1) they behave as *collective systems*; (2) they can infer subtle, unknown relationships from data; (3) they can generalize, meaning that they can respond correctly to patterns that are similar to the original training data; and (4) they are nonlinear, that is, they can solve some complex problems more accurately than linear techniques do. Amazingly, these characteristics have overcome all the constraints existing in LCA or BCA because they do not have any independence assumption among feature values. As a matter of fact, the four characteristics indeed specify exactly the desired behavior of CME.

However, Ho *et al.* [86] argued that it is inappropriate to use measurement values for measurement-level CME, because measurement values of correctly recognized samples usually overlap significantly with those of the wrongly recognized ones. Our observation also confirms this phenomenon. In fact, Ho's argument is only partially valid; our experiments [103, 104, 105, 106] have shown that through an effective data transformation function, original measurement values can be transformed into insignificantly overlapped ones. In the following two sub-sections, two issues are discussed: (1) What kinds of difficulties will a data transformation function encounter,



DT stands for data transformation

Figure 8: A new look for measurement-level CME.

and how can they be overcome? and (2) What objective should a data transformation function achieve? Then we present a family of useful data transformation functions.

5.3.1 Difficulties and Objectives of Data Transformation

In general, there are two incompatible physical meanings in measurement values supplied by various classifiers. The first is that the smaller the measurement is, the more probable that the corresponding class has the pattern. The second, on the contrary, is that the larger the measurement is, the more probable that the corresponding class has the pattern. For example, *distance* measurements belong to the first kind, and *similarity* and *confidence* measurements belong to the second. It is worthwhile to mention that even if the outputs of two classifiers have the same meaning, they may

be in quite differ ·nt scales; *e.g.* one may range from 0.0 to 1.0, the other from 100 to 10000. Because of the possibility of different meanings and scales, data transformation or normalization becomes essential before measurement data can be combined effectively.

The objective of data transformation is to convert the output of individual classifiers into a new form having the same meaning and scale. We call this new form "*likelihood*" measurement, which means that the possibility of a class having a pattern is proportional to the likelihood value, which ranges from 0.0 to 1.0. Let us consider first how to transform data into the same scale, and then into the same meaning.

5.3.2 Methods of Data Transformation

Suppose t_k^i is the transformed value of m_k^i through a transformation function T. In our opinion, an effective T should have the following desired properties:

1. The preference rank order will not be changed through data transformation, *i.e.* measurement values and their corresponding transformed values should have the same preference rank orders;

2. The range of the transformed values is 0.0 to 1.0, *i.e.* $0 \leq t_k^i \leq 1.0$;

3. The larger t_k^i is, the more likely the corresponding class i has the pattern;

4. The summation of all likelihoods produced from each classifier has a constant value. In convention, this constant is set to 1.0, *i.e.* $\sum_{i=1}^{M} t_k^i = 1.0$, where

$k \in \{1, \cdots, K\}$;

5. The data transformation function should be applicable to as many cases of measurements as possible. Ordinarily, from measurement values, one class will be specifically preferred to the other classes (*e.g.*, the smallest distance of one sample is 2.0, and at the same time all other distances of this sample are greater than 20.0). In Figure 9, those samples located in the kernel area of each class belong to this situation. However, there may also exist some cases in which no single class is especially preferred. This may happen in two situations:

   5a. The smallest distance is quite close to some other distances. Usually ambiguous patterns fall into this situation. For instance, the smallest measurement value is 3.5, and the others in ascending order are 3.7, 4.3, 15.3, 24.4, $\cdots$.

   5b. All measurement values (including the smallest one) are very large. This situation may occur when a classifier uses a *Mahalanobis*[1] - like distance calculation, and the class with the smallest distance happens to have a small feature variance. One example of such measurement values is {1436.2, 1466.4, 1498.9, 1640.1, $\cdots$}; here, again these values are listed in ascending order.

[1]Mahalanobis distance is measured by equation $(x - m)^t \sum^{-1} (x - m)$, where x is the feature vector of a pattern, m the mean feature vector of a class, and $\sum$ is the covariance matrix of the class

Situations **5a** and **5b** can be understood more easily through an example with a graphical illustration. For simplicity, in this example, suppose (1) there are only three classes, and their distributions are displayed in Figure 9; (2) the pattern space S can be divided into three kinds of areas: *kernel*, *normal*, and *ambiguous* areas; (3) there is a kernel area in each class, in which samples are not only densely populated, but also recognized easily and correctly; (4) the ambiguous area indicates the area in which the true identities of samples are hard to determine; and (5) normal area is the space which is neither a kernel nor an ambiguous area, in which although samples are sparsely populated, they can still be correctly recognized. In Figure 9, point **A** belongs to the situation of **5a**, and point **B** belongs to that of **5b**.

Considering all the desired properties of a transformation function, it is obvious that the conditional probability of Equation (24) is not suitable to apply here. For example, this equation cannot guarantee that preference rank orders are preserved; also the statement $\sum_{i=1}^{M} c_k^i(m_k^i) = 1.0$ is not always true. Furthermore, for the patterns belonging to situation **5b**, this equation is unable to compute meaningful confidence values. Therefore, a new transformation function should be developed; in fact, through our study, a family of data transformation functions has been proposed to serve this end. For the sake of clarity, the transformation functions for the first and second kinds of measurements will be described separately.

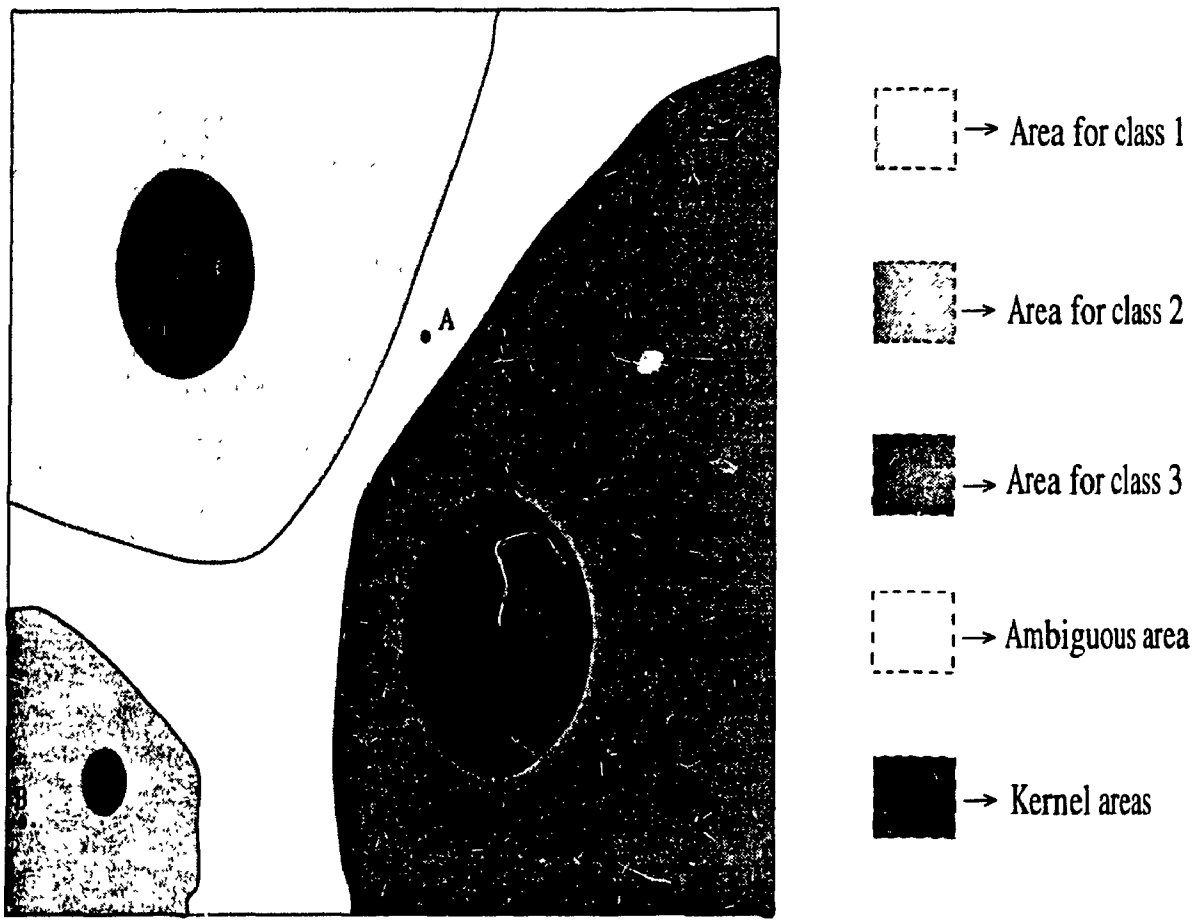


Figure 9: Distribution of measurement values for a 3-class domain.

First, we propose the transformation function T for the first kind of measurement:

$$T : m_k^i \longrightarrow t_k^i$$

and

$$t_k^i = \frac{\frac{1}{(m_k^i)^r}}{S_k}$$

and

$$S_k = \sum_{i=1}^{M} \frac{1}{(m_k^i)^r}.$$

where r is a real value and $r > 0$. The larger the value of r is, the larger the likelihood value of the first-choice class will be. When the value of r approaches infinitely large, the likelihood of the first-choice class becomes 1, and the likelihood of other classes become 0.

It is easy to verify that this proposed function can satisfy all of the above desired properties. For example, the summation of t_k^i over $i \in \{1, \cdots, M\}$ for classifier k equals 1.0, *i.e.* $\sum_{i=1}^{M} t_k^i = 1.0$. For the second kind of measurement, obviously the above T is not appropriate. However, since the reciprocal of the second kind of measurement is compatible with the first kind, function T can still be applied to the second kind, if measurement values have been inverted already. Therefore, for the second kind of measurement, the transformation function T becomes

$$T : m_k^i \longrightarrow t_k^i$$

where

$$t_k^i = \frac{(m_k^i)^r}{S_k},$$

and

$$S_k = \sum_{i=1}^{M} (m_k^i)^r.$$

Accordingly, both kinds of measurements can assume one form, with the same meaning (the larger the better) and scale ([0.0 - 1.0]). The value of r will be decided from experiments so that the best recognition performance can be produced.

5.3.3 A Multi-layer Perceptron

Since the ground-truth class of each learning pattern is known already, it is a *supervised learning* process. Accordingly, a multi-layer perceptron (MLP) with the *Generalized Delta Rule* (GDR) becomes a good choice to serve the combination function E, because it has been used successfully in various pattern recognition applications with good recognition results.

As a matter of fact, the radial-basis function (RBF) network is another suitable alternative for function E. Figure 10 shows the basic structure of the RBF network, which is a three-layer network containing only one hidden layer. Each hidden node is a processing unit which performs a radial-basis function B. The most popular and widely used radial-basis function is the Gaussian function $B(d) = exp(-\frac{d^2}{\rho^2})$, which depends on the distance $d = \|x - c\|$ (where $\|\cdot\|$ denotes a vector norm) between the input vector x and the center c, and ρ is a bandwidth parameter. Geometrically,

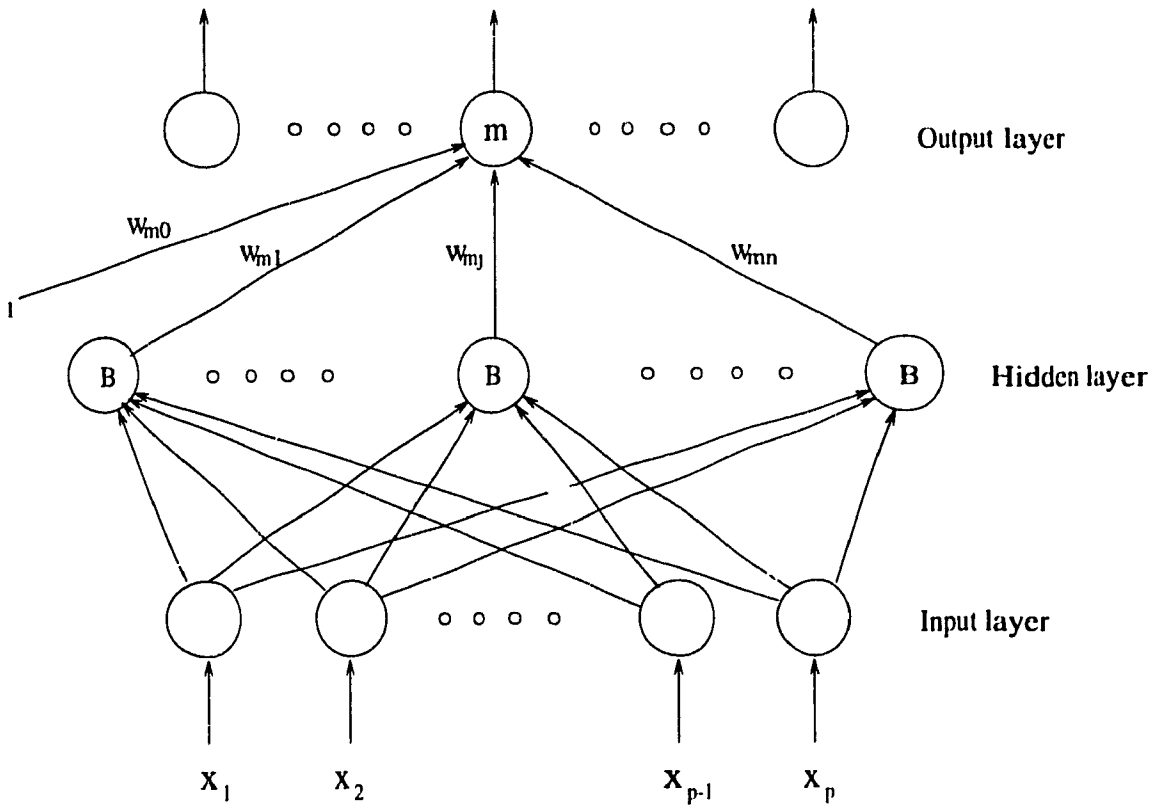


Figure 10: A basic structure of the RBF network.

a Gaussian function has a peak at the center c, and decreases monotonically as the distance d from the center increases. Each output node simply performs a linear weighted summation of all basis functions of the hidden layer. Therefore, the transformation from the input space to the hidden-unit space is *nonlinear*, whereas the transformation from the hidden-unit space to the output space is *linear*. In an RBF network the centers and bandwidths of radial-basis functions are usually fixed (*i.e.* they will not be changed during the learning process) and only the connection weights between the hidden and output layers will be updated. Therefore, the learning process of the RBF network is to adjust the connection weights between the hidden and output layers with a *delta rule* so that the network can obtain its best classification for a given training data set. In the literature [107, 108], the RBF network has been reported to possess desirable properties on network training, generalization and garbage rejection. However, to pursue this research, MLP is chosen to integrate the transformed measurement values simply because it is easier to implement[2].

Often, MLP is called the Back-Propagation (BP) network since during net training, the error caused by the difference between the desired output vector and the output layer's response to an input vector propagates back through connections between layers and adjusts appropriate connection weights so as to minimize the error.

[2]To implement an RBF, three parameters should be decided beforehand: (1) the number of centers, (2) the locations of centers, and (3) the bandwidths of Gaussian functions.

This learning ability by propagating error backward enables the net to learn the mapping between input vectors and desired output vectors automatically. Therefore, for recognition, when the input vector of an unlabeled pattern is presented to the input layer, the output layer then tends to match with the desired output vector. The significant contribution of the back-propagation algorithm is that it can form arbitrarily complex decision regions in the input space by only using a three-layer perceptron [35]. The decision region of a certain class may contain several sub-regions, which need not be connected. After decision regions are constructed, the input pattern is classified into the class whose decision region is sufficiently close to this pattern. Mapping from input vectors to class decision regions enables BP to achieve great success in solving various problems. For completeness, the following will describe (1) the system architecture of a MLP, and (2) the generalized delta learning algorithm (GDR).

5.3.4 Network Architecture

Figure 11 shows a three-layer network with one input layer, one hidden layer and one output layer, where the transformed measurement values are fed to the input layer and $O_1, \ldots,$ and O_M are the output values from the output layer. In general, an N-layer network ($N \geq 2$) has one input layer, one output layer and $N-2$ hidden layers. The nodes of the input layer are called input nodes, those of hidden layers hidden nodes, and those of the output layer output nodes. Except for the input

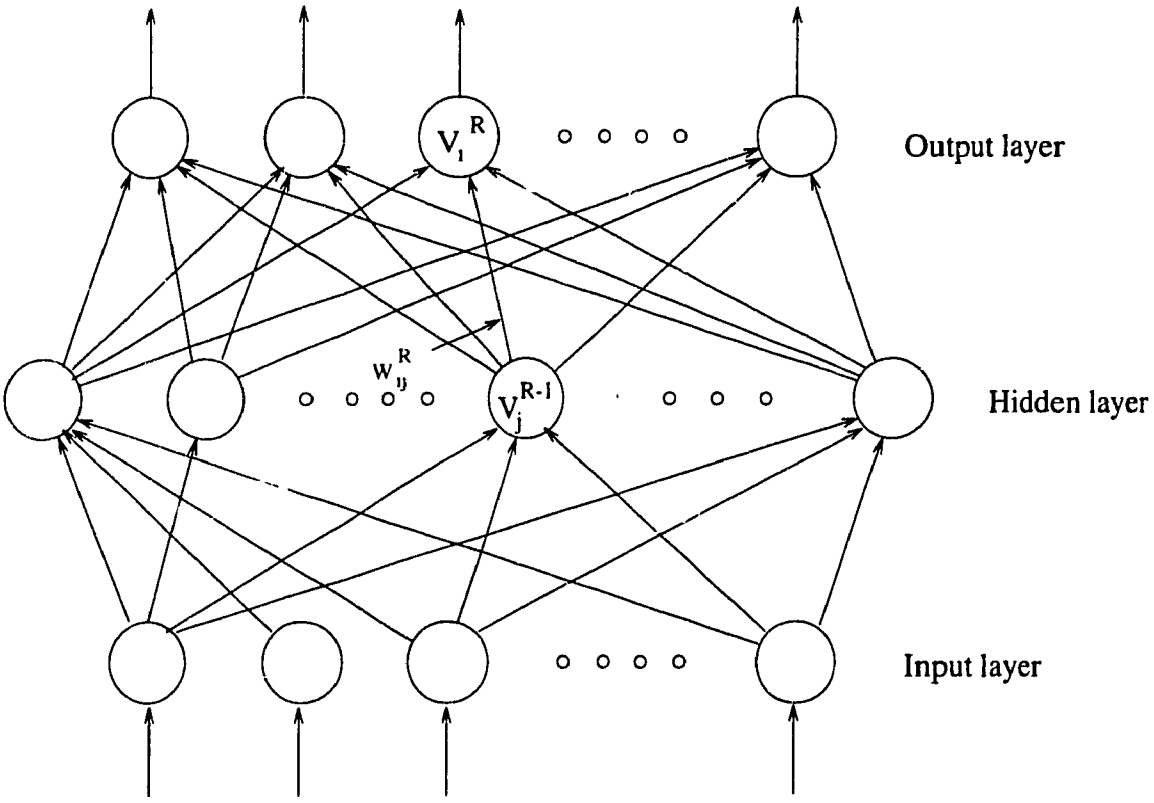


Figure 11: Diagram of a three-layer feedforward network.

nodes, the net input to each node is the sum of the weighted outputs of the nodes in the prior layer, and the output of this node is the value that the weighted sum passes through a nonlinear activation function f. Usually, the activation function is a sigmoidal function defined as

$$f(x) = \frac{1}{1 + \exp^{-(x+\theta)}} \tag{38}$$

where θ is a bias. The effect of a positive θ is to shift the activation function to the left along the horizontal axis. Obviously, the value of the activation functio.. f varies from 0.0 to 1.0. For CME, the number of input layer nodes equals that of the total transformed measurement values, and the number of output layer nodes equals the total number of M classes.

5.3.5 The Generalized Delta Rule

The Generalized Delta Rule (GDR) is an *iterative gradient algorithm* designed to minimize the *mean square error* between the actual and desired outputs of a MLP. Through a set of learning samples, it can find the best weights w_{ij} automatically, enabling this network to exhibit optimal classification ability. Since the error is propagated from the last layer (*i.e.* the output layer) backward to the first layer (*i.e.* the input layer), the weight updating algorithm is called the Back-Propagation Algorithm (BPA) as well. Suppose there is an N-layer perceptron, V_j^r denotes the jth node of the rth layer, and w_{ij}^r denotes the connection weight between the ith node of the rth

layer and the jth node of the $(r-1)$th layer. Then, the back-propagation training algorithm is implemented as follows:

Step 1: Initialize all weights and offsets with small random values.

Step 2: Present an input vector I and a desired output vector O. Apply I to the input layer ($r = 0$) so that $V^0 = I$.

Step 3: For other layers, namely $r = 1, \cdots, N$, perform forward computation:

$$V_i^r = f(\sum_j w_{ij}^r V_j^{r-1})$$

where w_{ij}^r represents the connection weight from V_j^{r-1} to V_i^r.

Step 4: Compute the errors for the output layer:

$$\delta_i^R = V_i^R = V_i^R(1 - V_i^R)(O_i - V_i^R)$$

Step 5: Compute the back-propagation errors for preceding layers $N-1, \cdots, 1$:

$$\delta_j^{r-1} = V_j^{r-1}(1 - V_j^{r-1}) \sum_i w_{ij}^r \delta_i^r$$

Step 6: Adjust all weights:

$$w_{ij}^r(t+1) = w_{ij}^r(t) + \eta \delta_j^r V_j^{r-1} + \alpha \Delta w_{ij}^r(t) \tag{39}$$

where η is the learning rate, α is a constant momentum coefficient and $\Delta w_{ij}^r(t) = w_{ij}^r(t) - w_{ij}^r(t-1)$. The third term of Equation (39) is used to specify that the

change in w_{ji}^r at the $(t+1)$th iteration should be somewhat similar to the change undertaken at the tth iteration. In this way some inertia is built in, and momentum in the rate of change is conserved to some degree. Thresholds are adjusted in a way similar to weights.

Step 7: Repeat by going to step 2.

The algorithm continues until the iteration number is larger than a pre-specified threshold number, or the overall error (which is the mean square difference between the desired and the actual output vectors for all training patterns) is reduced to an acceptable level.

5.3.6 The Decision Rule

After weight training, for the recognition of a testing sample x, the values of the output layer will generally fall into one of the three situations described below:

1. x is classified: only one output node is active, which represents the recognized class of x;
2. The net is confused about the class of x: more than one output node is active;
3. The net fails to make a decision as to the class of x: none of the output nodes is active.

Considering these three situations, the final decision rule is defined as

$$E(x) = \begin{cases} j & \text{, if } O_j = Max_1 \text{ and } (Max_1 - Max_2) \geq \alpha; \\ M+1 & \text{, otherwise.} \end{cases} \tag{40}$$

where α is a threshold, $Max_1 = max_{i \in \Lambda} O_j$, and $Max_2 = max_{i \in \Lambda - \{j\}} O_j$. When α is large, only samples of situation 1 will be recognized decisively, and samples of the other two situations will be rejected.

5.4 Improvement Strategies for Multi-Layer Perceptrons

It is well known that *training time* and *generalization ability* are two main concerns related to neural networks. In our system, three strategies [103, 104] have been proposed to improve system performance in both *speed* and *recognition accuracy*. In general, these strategies are useful in dealing with over-training and a long-period of computation time for network training.

5.4.1 Training by Boundary Samples

The major problem of MLP is that generally it requires a long period for weight training (*e.g.* two whole days using a SUN-4 workstation at 55 MIPs). Originally, in the training stage, every sample will be used to back-propagate its error to update weights. This means that whether or not a sample is well classified, the same amount of time will be spent on it to update weights. In fact, this is not an efficient approach to weight training, because only the samples on the boundaries or wrongly recognized

are really useful for weight updating. Accordingly, this strategy modifies the learning process for each sample as (1) it computes the outputs of the network forward, and (2) it updates weights backward with GDR if this sample is wrongly or hardly recognized; otherwise this sample has no effect on changing weights at all. Experiments have shown that this strategy can bring about a 5-fold speed-up, and at the same time it is capable of maintaining the recognition accuracy.

5.4.2 Training by Partition

Intuitively, if the entire set of training samples is divided into several partitions, and each partition can have its own appropriate combination approach, then a better recognition should be achievable. For example, for one partition, by using MLP the distribution of this partition can be better described if the corresponding weights are derived from this partition alone. Therefore, each partition will have a higher recognition rate than it would if weights were generated from the whole set of training samples. Accordingly, CME with partition will produce higher accuracy than without partition. From the implementation point of view, it is important that the selecting criteria for partition must be easily realized. One method to do this is by measuring the *quality* of the processed images, which can be estimated from the decisions of individual classifiers. Heuristically, if one pattern is easily recognizable by classifiers, then this pattern has a high probability to be a regular and good-quality writing pattern. On the contrary, if one is difficult to recognize, then it tends to

reflect poor-quality writing. For example, in an experiment, we divided the training set (19,353 samples) into two subsets (A and B). Set A contains 10,411 good patterns, while Set B contains 8,942 poor patterns. A pattern is considered to be of good quality only when all three classifiers successfully recognize it, and for each classifier the difference between the largest and second largest transformed values is greater than 0.15; otherwise, this pattern is considered to be of poor quality. Because the entire pattern space is partitioned into two subsets, correspondingly there are two multi-layer perceptrons as well, one for each subset. Due to a smaller number of training samples, each perceptron can be trained fairly quickly. For testing, an unlabeled pattern is first distinguished into one of two subsets (good- or poor-quality) by the same criterion used for partitioning the training set, and then is recognized by its corresponding net. Among the total testing set, 11,159 and 10,865 samples are considered as good- and poor-quality samples, respectively. The experiment shows that except for 4 samples, all good-quality samples are correctly recognized (*i.e.* 99.96% recognition accuracy), and the average recognition rate of poor-quality samples is 94.36%. Accordingly, the average recognition rate of the total testing set is 97.20%. For comparison, the average recognition rate of the testing set by a single multi-layer perceptron trained by the entire learning set is 97.05%. Interestingly, a majority voting algorithm also produce a 99.96% recognition accuracy to the good-quality samples. Therefore, to further speed up the recognition process, voting will serve

the CME for the good-quality subset, and for the poor-quality subset the multi-layer perceptron will be used.

5.4.3 Weight Reduction

Equation (38) shows that $f(\mathbf{X}, \mathbf{W}, \theta) = \frac{1}{1+\exp^{-(\mathbf{W}\mathbf{X}+\theta)}}$. Thus, the derivative of f with respect to X becomes

$$\frac{\partial f}{\partial X} = \mathbf{W} * f * (1 - f). \tag{41}$$

This equation shows that the derivative of f with respect to $\mathbf{X}$ is influenced by three parameters: $\mathbf{W}$, f, and $1 - f$. Heuristically, if there is a way to keep the recognition rate of the training set the same, it is desired that the value of this derivative be small, since the activation function is more stable for slight perturbations of input $\mathbf{X}$. Because both f and $1 - f$ contain parameter $\mathbf{W}$, the reduction of the norm of $\mathbf{W}$ seems to be the most effective factor for stabilizing $\frac{\partial f}{\partial X}$. In general, there are two ways to reduce the norm of $\mathbf{W}$. The first is applied during the weight training stage, by adding a certain amount of penalty with respect to the norm of $\mathbf{W}$ to the total error function [109, 110]. The second is applied after the weight training process, by reducing the norm of $\mathbf{W}$ if the recognition rate of the reduced $\mathbf{W}$ is not lower than the original one. In this experiment, we adopt the second approach, which contains the following steps:

Step 1: use GLVR to derive $\mathbf{W}$ and R, the average recognition rate of the training samples;

Step 2: reduce the norm of weights by a certain ratio λ ($0 < \lambda < 1$), *i.e.*

$\mathbf{W}' := \lambda * \mathbf{W}$;

Step 3: compute the recognition rate R' with respect to $\mathbf{W}'$ for the whole training set;

Step 4: IF $R' >= R$,

THEN

$\mathbf{W} := \mathbf{W}'$,

$R := R'$, and

GOTO Step 2;

ELSE stop.

After the algorithm stops, $\mathbf{W}$ is the final weight matrix. In practical applications, the weight reduction can be implemented by the following three levels:

1. The *net* level: all weights in the net are reduced by the same λ,

2. The *layer* level: the weights of the same layer are reduced by the same λ, but the weights of the other layers are kept constant.

3. The *connection* level: a single weight is reduced by λ and at the same time all the other weights are kept constant.

As a matter of fact, in our implementation, weight reduction at the connection level can not only reduce but also enlarge individual weights. In other words, weights are updated in both directions: either reduction or enlargement. However, reduction will be tried first. Interestingly, GDR updates all weights during each iteration; but at the connection level of weight reduction, weights are changed in different iterations. Accordingly, these three levels of weight reduction complement GDR, providing MLP with a more powerful learning capability and producing a higher accuracy.

Chapter 6

Experiments

6.1 Introduction

In Chapters 3, 4, and 5, we have introduced many combination approaches, including our own. For abstract-level classifiers, although the BKS method has been proven to be the best combination function from the statistical point of view, we are curious whether it can produce the highest recognition accuracy for the practical cases as well. For measurement-level classifiers, we also want to know what degree of accuracy our combination models can achieve, and whether our models can outperform other measurement-level combination functions. Section 6.2 describes the data used in this experiment. Section 6.3 describes the experiments performed for abstract-level CME, and Section 6.4 the experiments for measurement-level CME. Both experiments are

performed on ITRI's[1] numeral database, which consists of 46,451 samples.

6.2 Description of the Experimental Data

The data used for this experiment come from ITRI's numeral data base, which contains 46,451 samples collected from more than 1,000 persons. Each person wrote a complete set of numerals (*i.e.* $0, 1, \cdots, 9$) about 5 times; no writing constraint was imposed for each numeral except a bounding box. Because of the large number of participants, this data base approximately reflects the daily writing styles of numerals in Taiwan. Figure 12 shows some samples of this database. The whole data collection was divided into 10 sets. Each set contains various numbers of samples for different numerals. The first set (5,074 samples) was used for training individual classifiers, and the other 9 sets (41,377 samples) for testing their performances. Table 4 lists the total number of samples, and the total number of samples of each class for every individual set. Three classifiers (called e_1, e_2 and e_3) developed by CENPARMI[2] and ITRI OCR research teams were chosen as three experts in the following experiments. The first uses gradient features [111], the second loci features [112], and the third peripheral features [113]. For one pattern, each classifier produces 10 measurement values; therefore, there are in total 30 measurement values for each pattern. Table 5 shows the respective classification performances of the three classifiers on the testing

[1]ITRI is a government-sponsored research institute in Taiwan.
[2]CENPARMI is Centre for Pattern Recognition and Machine Intelligence at Concordia University.

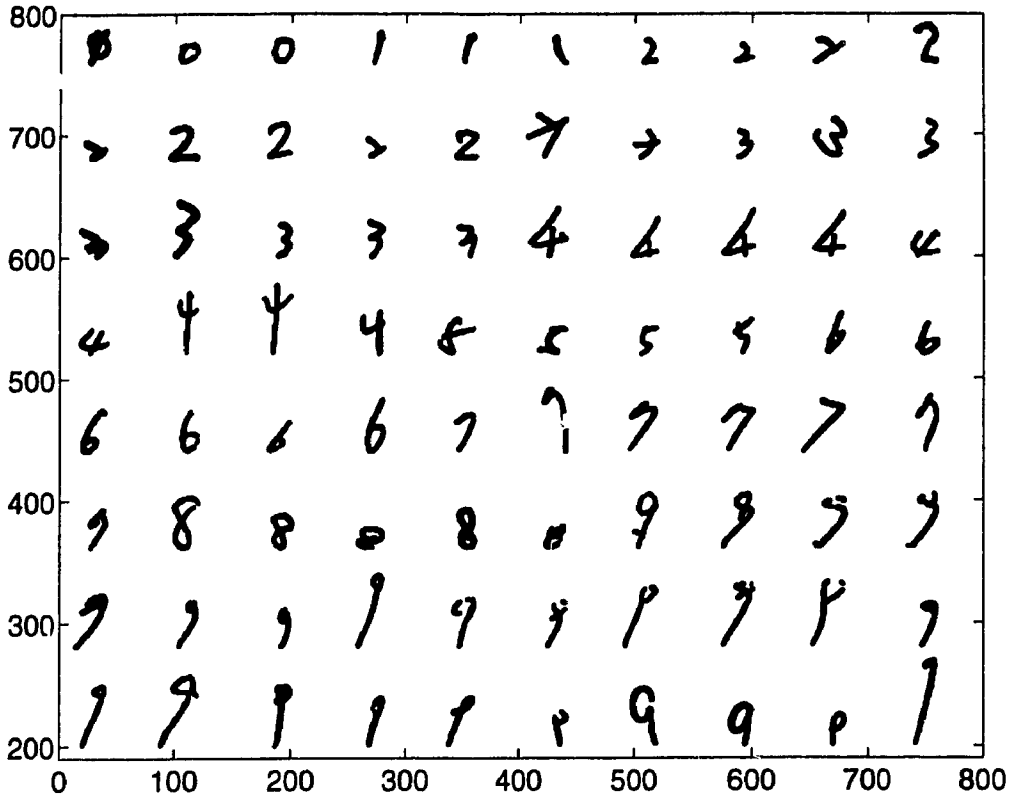


Figure 12: 80 samples from ITRI's numeral database.

sets, where Rec., Sub., and Rel. denote the recognition and substitution rates and the reliability, respectively.

6.3 Experiments on Abstract-Level CME

There are four purposes for performing these experiments: (1) to investigate and compare the performance of the BKS method with other three abstract-level

	0	1	2	3	4	5	6	7	8	9	total
set 0	554	597	613	585	452	456	452	492	421	452	5074
set 1	520	572	562	546	427	444	436	478	418	443	4846
set 2	488	575	584	545	418	439	420	481	403	415	4768
set 3	504	636	569	574	435	468	403	505	424	486	5004
set 4	488	570	590	519	422	422	409	480	413	422	4735
set 5	430	572	600	529	463	410	399	447	357	404	4611
set 6	363	511	505	467	421	381	373	451	349	364	4185
set 7	451	575	562	491	397	407	391	436	376	390	4476
set 8	354	515	526	484	420	386	395	420	380	368	4248
set 9	451	550	548	502	413	412	406	432	393	397	4504

Table 4: Total number of samples and total number of samples of each class in each set of the ITRI's numeral database.

	Rec.	Sub.	Rel.
e_1	90.37	9.63	0.9037
e_2	90.93	9.07	0.9093
e_2	92.14	7.86	0.9214

Table 5: Recognition performances of individual classifiers using 41,377 testing samples (sets 1 - 9).

CME methods, voting, Bayesian and Dempster-Shafer; (2) to ascertain that the BKS method will not degenerate its performance even when there is strong dependence among classifiers; (3) to examine the validity of the semi-monotonicity property by using the BKS method; and (4) to see whether consecutive CME can really improve the final classification performance or not. The following experiments will reveal the differences among the four CME methods, based on the same experimental environment (*i.e.* the same classifiers and the same learning and testing data). For this experiment, only the first-choice class label supported by each classifier is taken into consideration, so that abstract-label information may be produced.

6.3.1 Experiments with Re-Substitution Estimation

The goal of this experiment is to compare the performances of all four CME methods in the context that fully representative learning samples are provided so that the classifiers have the same classification behavior on both training and testing data. To simulate this situation, all 41,377 samples (sets 1-9) were used for both learning and testing; it is a re-substitution estimation [13]. Tables 6(a – d) show the results produced by the voting, Bayesian, Dempster-Shafer, and BKS methods with different thresholds, respectively. In these tables, α is a threshold value which controls the reliability of decision making; the greater that α is, the more reliable that decision is made. Figure 13 shows the recognition performances of the four combination functions in graph representation. Obviously, the BKS method performs best; its recognition

rate can achieve 96.21% with no error. This shows from the statistical point of view that the optimality of the BKS method is guaranteed, and CME can considerably improve recognition performance (with no rejection, the best recognition rate among the three individual classifiers is 92.14%). However, in practice, it is improper to include training samples for testing. Therefore, the high performance of the BKS method is not practical; it may come from over-adaptation of the BKS, because all learning samples are also used for testing. The next experiment was performed to investigate the real classification results of the four CME methods with the same experimental data.

6.3.2 Experiments with Leave-One-Out Estimation

This experiment adopted a leave-one-out estimation [13], so that an unbiased comparison can be performed. With leave-one-out estimation, all samples but one are learned, and then the unlearned one is tested; the same operation is repeated until every sample in the data set has been left out and tested. Experimentally, the leave-one-out estimation has been found to be approximately unbiased for any data distribution. However, usually it suffers from extremely excessive computation, as it computes the distribution for each repeated run. Fortunately, the Bayesian, Dempster-Shafer, and BKS methods all store their information in matrices, thus only a small portion of those matrices should be updated in each run. Therefore, all four methods can be efficiently implemented for this estimation. For example, suppose in

α	Rec.	Sub.	Rej.	Rel.
0.000	93.92	6.08	0.00	0.9392
0.333	93.92	6.08	0.00	0.9392
0.667	93.24	4.03	2.73	0.9586
1.000	82.09	0.77	17.14	0.9908

(a) The voting method

α	Rec.	Sub.	Rej.	Rel.
0.000	95.17	4.83	0.00	0.9517
0.300	94.62	4.42	0.96	0.9554
0.500	94.23	3.81	1.95	0.9611
0.700	93.55	3.03	3.42	0.9686
0.900	92.27	2.40	5.33	0.9747
0.990	86.89	1.16	11.95	0.9868
0.999	83.08	0.80	16.12	0.9905

(b) The Bayesian method

α	Rec.	Sub.	Rej.	Rel.
0.000	94.13	5.87	0.00	0.9413
0.100	93.24	4.03	2.73	0.9586
0.750	93.24	4.03	2.73	0.9586
0.800	90.50	3.20	6.30	0.9659
0.850	87.04	2.18	10.77	0.9755
0.900	82.09	0.77	17.14	0.9908
0.990	82.09	0.77	17.14	0.9908

(c) The Dempster-Shafer method

α	Rec.	Sub.	Rej.	Rel.
0.000	96.21	3.79	0.00	0.9621
0.100	95.63	3.12	1.26	0.9685
0.300	94.92	2.61	2.46	0.9732
0.400	94.06	2.13	3.80	0.9778
0.500	93.18	1.76	5.05	0.9814
0.800	90.46	1.09	8.45	0.9881
0.950	85.15	0.78	14.07	0.9909

(d) The BKS method

Table 6: Results for 41,377 samples using a re-substitution estimation to combine the three classifiers (e_1, e_2 and e_3).

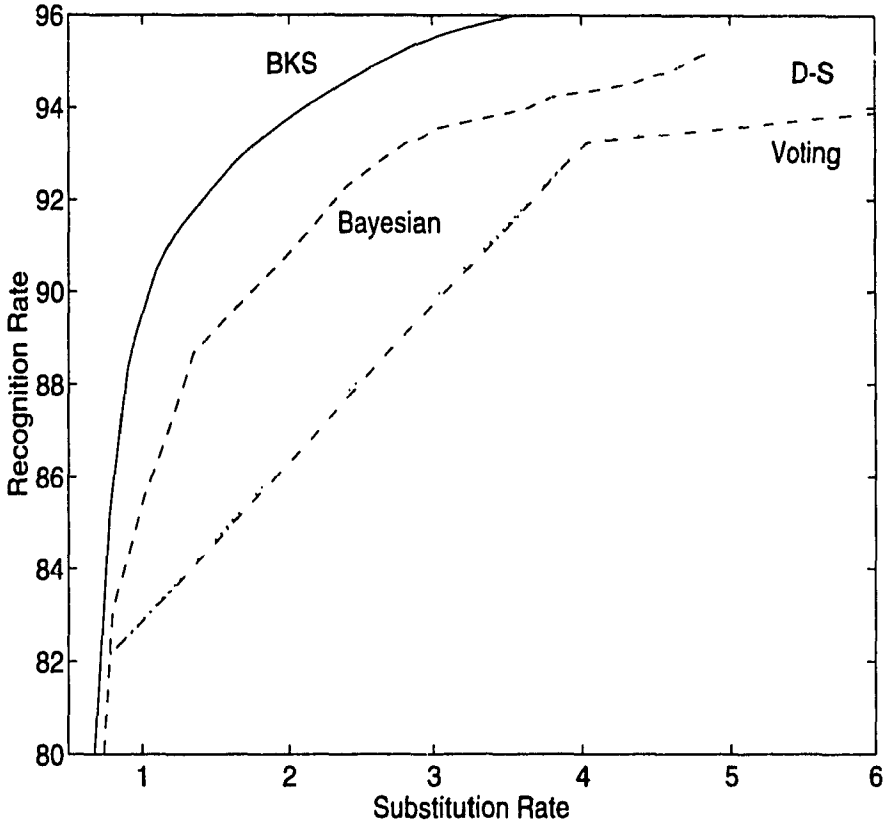


Figure 13: Graphic representation of the recognition performances of four CME methods by using a re-substitution estimation, where "D-S" represents "Dempster-Shafer".

total there are N samples (*i.e.* $x_1, \cdots, x_N$) and $I(x)$ stands for the ground-truth class label of pattern x; then the BKS method can be executed according to the following procedure:

Step 1: use all training samples to construct a BKS;

Step 2: set $i = 0$;

Step 3: remove the information of x_i from the constructed BKS and compute the most representative class label with respect to the current FC of x_i. Three operations are performed in this step:

a. $n_{e_1(x_i)\cdots e_K(x_i)}(I(x_i)) := n_{e_1(x_i)\cdots e_K(x_i)}(I(x_i)) - 1$;

b. $T_{e_1(x_i)\cdots e_K(x_i)} := T_{e_1(x_i)\cdots e_K(x_i)} - 1$;

c. compute $R_{e_1(x_i),\cdots,e_1(x_i)}$ according to Equation (5);

Step 4: with $e_1(x_i), \cdots, e_K(x_i)$ and $I(x_i)$, according to Equation (6), make the classification decision $E(x_i)$, then check whether the decision equals $I(x_i)$ or not;

Step 5: add the information of x_i into the constructed BKS; two operations are performed in this step:

$n_{e_1(x_i)\cdots e_K(x_i)}(I(x_i)) := n_{e_1(x_i)\cdots e_K(x_i)}(I(x_i)) + 1$;

$T_{e_1(x_i)\cdots e_K(x_i)} := T_{e_1(x_i)\cdots e_K(x_i)} + 1$;

Step 6: $i := i + 1$;

IF $i > N$ THEN Go to Step 3;

ELSE stop.

With a leave-one-out estimation, Table 7 and Figure 14 show the tabular and graphical representation of the corresponding recognition performances. Although the BKS method produces the highest recognition rate when the rejection rate is low, it performs poorly when the rejection rate is high. This phenomenon results from no repetition of rare or irregular cases, as illustrated in Section 4.7. To overcome this situation, two approaches have been already proposed and discussed in Section 4.7. The first is simply to collect more training samples. It is obvious that if there are more samples to make the irregular cases repeatable, then the substitution rate should be more sensitive to the threshold α and more easily reduced. The second is to initialize the number of incoming samples of each cell to 1 instead of 0. To verify the first approach, a simulation is performed by presenting the 41,377 samples twice. Table 8 lists the corresponding recognition performance, which indeed shows an improvement of the recognition performance for the area with high rejection rates. With the second approach, the classification is performed and its result is listed in Table 9. To have a clear visual comparison, Figure 15 shows the recognition performance of all four combination methods, but now the BKS method is modified by the second approach. From this figure, it is easy to observe that the BKS method with the second approach produces the best result in almost every situation among the four measurement-level combination methods. Only when the substitution rate is low, seemingly it performs equally with the Bayesian method. Since the second approach can be implemented

α	Rec.	Sub.	Rej.	Rel.
0.000	93.92	6.08	0.00	0.9392
0.353	93.92	6.08	0.00	0.9392
0.667	93.24	4.03	2.73	0.9586
1.000	82.09	0.77	17.14	0.9908

(a) The voting method

α	Rec.	Sub.	Rej.	Rel.
0.000	95.14	4.86	0.00	0.9514
0.500	94.19	3.82	1.98	0.9610
0.700	93.54	3.05	3.41	0.9684
0.900	92.25	2.49	5.25	0.9737
0.950	90.65	1.97	7.38	0.9787
0.980	88.63	1.34	10.02	0.9851
0.999	83.08	0.80	16.12	0.9905

(b) The Bayesian method

α	Rec.	Sub.	Rej.	Rel.
0.000	94.13	5.87	0.00	0.9413
0.100	93.24	4.03	2.73	0.9586
0.400	93.24	4.03	2.73	0.9586
0.700	93.24	4.03	2.73	0.9586
0.800	90.50	3.20	6.30	0.9659
0.850	87.04	2.18	10.77	0.9755
0.900	82.09	0.77	17.14	0.9908

(c) The Dempster-Shafer method

α	Rec.	Sub.	Rej.	Rel.
0.000	95.31	4.36	0.33	0.9563
0.100	95.03	3.39	1.58	0.9655
0.300	94.39	3.02	2.60	0.9690
0.500	92.65	2.14	5.21	0.9775
0.700	90.84	1.60	7.57	0.9827
0.950	84.82	1.15	14.03	0.9866
0.980	66.54	0.77	32.69	0.9886

(d) The BKS method

Table 7: Results for 41,377 samples using a leave-one-out estimation to combine the three classifiers (e_1, e_2 and e_3).

without extra cost and the resulting performance is actually better than the original BKS method, it is adopted in the following experiments. This means that whenever the BKS method is mentioned in later discussion, it is the BKS method modified by the second approach. Also, the following experiments are based on the leave-one-out estimation.

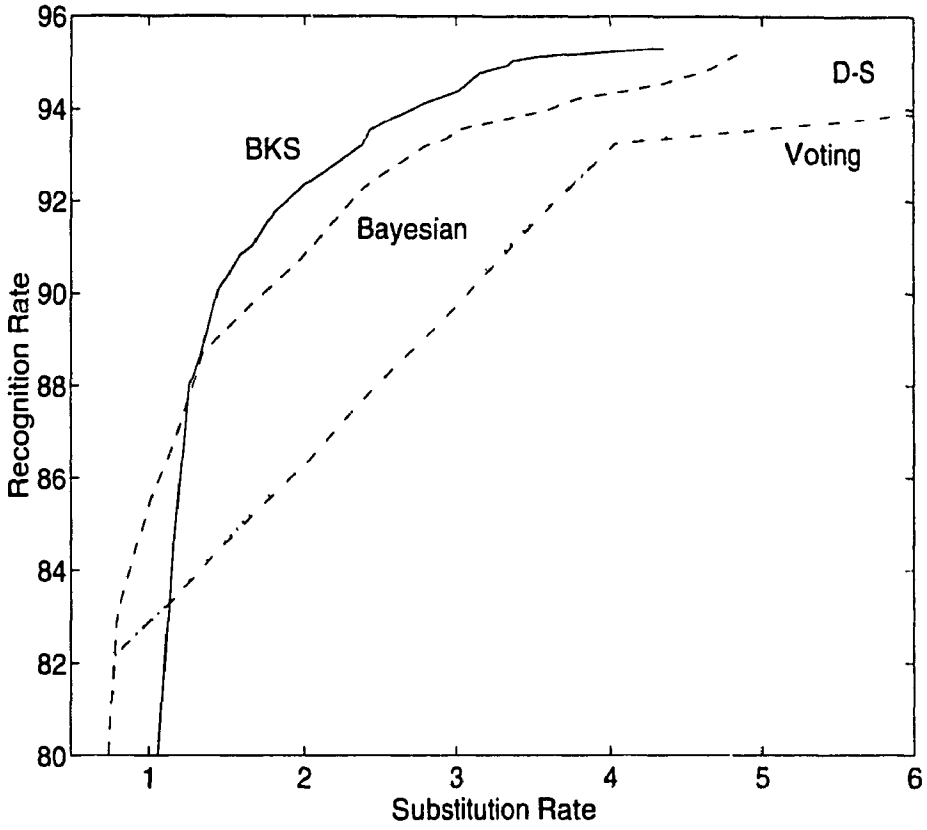


Figure 14: Graphic representation of the performances of four CME methods by using a leave-one-out estimation, where "D-S" represents "Dempster-Shafer".

α	Rec.	Sub.	Rej.	Rel.
0.000	95.77	4.23	0.00	0.9577
0.100	95.57	3.33	1.10	0.9663
0.300	94.76	2.85	2.38	0.9708
0.400	93.98	2.24	3.78	0.9767
0.500	93.02	1.82	5.16	0.9808
0.700	91.17	1.32	7.51	0.9857
0.900	88.34	0.94	10.72	0.9894
0.950	85.15	0.80	14.05	0.9907

Table 8: Recognition results of repeating sets 1 to 9 twice by using the BKS method and a leave-one-out estimation to combine the three classifiers (e_1, e_2 and e_3).

α	Rec.	Sub.	Rej.	Rel.
0.000	95.31	4.36	0.33	0.9563
0.050	95.03	3.47	1.50	0.9648
0.200	94.08	2.72	3.20	0.9719
0.300	92.97	2.28	4.76	0.9761
0.400	91.60	1.83	6.56	0.9804
0.500	90.42	1.44	8.14	0.9844
0.600	89.01	1.11	9.88	0.9876
0.950	82.09	0.77	17.14	0.9908

Table 9: Results for 41,377 samples using the BKS method and a leave-one-out estimation to combine the three classifiers (e_1, e_2 and e_3), where in each cell, the number of incoming samples of each class is initially set to 1.

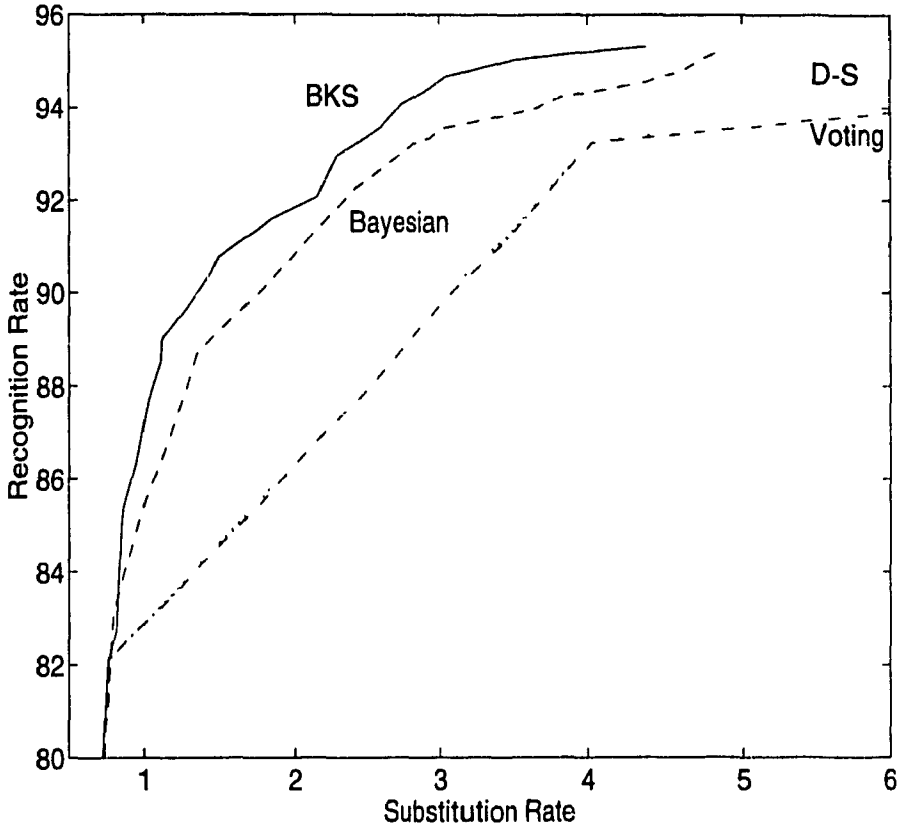


Figure 15: Graphic representation of the performances of four CME methods by using a leave-one-out estimation, where BKS stands for the modified BKS method, which initializes the number of incoming samples of each class in each cell to 1 instead of 0.

Interestingly, all four performance curves are close together in the lower-left corner, and intersect near the point with 82.09% recognition and 0.77% substitution. This special performance indeed is the performance of the voting method with a threshold $\alpha = 1.0$. Therefore, we know that even when all three classifiers give their decisions to the same class label, still there are 0.77% of the tested samples which are misrecognized. This observation enables us to state that if the required recognition rate is higher than 82.09% and the required substitution rate is lower than 0.77%, then it is impossible to achieve this goal by using only the abstract-level information of these three classifiers.

6.3.3 The Combination of Dependent Classifiers

The goal of this experiment is to compare the results of the four combination methods in the context of strong dependence among classifiers. To simulate such strongly dependent classifiers, e_1 was used twice (named e_1 and e_1') to combine with e_2 and e_3. Therefore, there are, in total, four classifiers (e_1', e_1, e_2, and e_3). By a leave-one-out estimation, Table 10 and Figure 16 show the results of these four methods in tabular and graphic forms, respectively. Again, we can easily verify that (1) the BKS method maintains the best performance among them, and (2) the performance of the BKS method in Table 10(d) is exactly the same as that shown in Table 9. As for the voting, Bayesian and Dempster-Shafer methods, the degraded performances in Tables 10(a), 10(b) and 10(c) show that they were affected significantly due to the

undesirable dependence among classifiers.

6.3.4 The Semi-Monotonicity Experiment

The purpose of this experiment is to investigate the validity of the semi-monotonicity property of the BKS method, that is whether equal or even better recognition performance can be achieved by combining more classifiers. Since there are only three classifiers available, we will check whether the CME performance of the three classifiers is better than that of any two classifiers, and whether the CME performance of any two classifiers is better than that of any single one. Suppose MAX(n) and MIN(n) stand for the highest and lowest performances that the BKS method can achieve with n classifiers. To verify the semi-monotonicity property of the three classifiers, the hypotheses that MIN(3) $\leq$ MAX(2) and MIN(2) $\leq$ MAX(1) must be true. Table 11 lists the performances of all the possible combinations. It shows MAX(1) = 92.14%, MAX(2) = 94.34% , MIN(2) = 94.08% , and MIN(3) = 95.31%, therefore MIN(3)=95.31% > MAX(2)=94.34% and MIN(2)=94.08% > MAX(1)=92.14%. This concludes that the semi-monotonicity property is preserved in this experiment.

6.3.5 Experiment on Consecutive CME

As mentioned before, consecutive CME regards combination functions as classifiers. Accordingly, consecutive CME combines not only individual classifiers but also

a	Rec.	Sub.	Rej.	Rel.
0.000	92.02	7.98	0.00	0.9202
0.250	92.02	7.98	0.00	0.9202
0.500	89.34	4.29	6.37	0.9542
0.750	88.29	2.61	9.10	0.9712
1.000	82.09	0.77	17.14	0.9908

(a) The voting method

a	Rec.	Sub.	Rej.	Rel.
0.000	94.65	5.35	0.00	0.9465
0.200	94.41	5.16	0.44	0.9482
0.600	93.44	4.36	2.20	0.9554
0.800	92.37	3.60	4.02	0.9625
0.900	90.77	3.13	6.10	0.9666
0.950	90.42	2.56	7.02	0.9725
0.990	88.80	1.73	9.47	0.9809
0.999	85.62	1.11	13.26	0.9872

(b) The Bayesian method

a	Rec.	Sub.	Rej.	Rel.
0.000	94.29	5.71	0.00	0.9429
0.200	89.34	4.29	6.37	0.9542
0.400	89.34	4.29	6.37	0.9542
0.600	89.34	4.29	6.37	0.9542
0.800	88.29	2.61	9.10	0.9712
0.900	88.29	2.61	9.10	0.9712
0.950	88.29	2.61	9.10	0.9712
0.980	85.55	1.78	12.66	0.9796

(c) The Dempster-Shafer method

a	Rec.	Sub.	Rej.	Rel.
0.000	95.31	4.36	0.33	0.9563
0.050	95.03	3.47	1.50	0.9648
0.200	94.08	2.72	3.20	0.9719
0.300	92.97	2.28	4.76	0.9761
0.400	91.60	1.83	6.56	0.9804
0.500	90.42	1.44	8.14	0.9844
0.600	89.01	1.11	9.88	0.9876
0.950	82.09	0.77	17.14	0.9908

(d) The BKS method

Table 10: Recognition results for 41,377 samples by combining the four classifiers (e_1', e_1, e_2 and e_3).

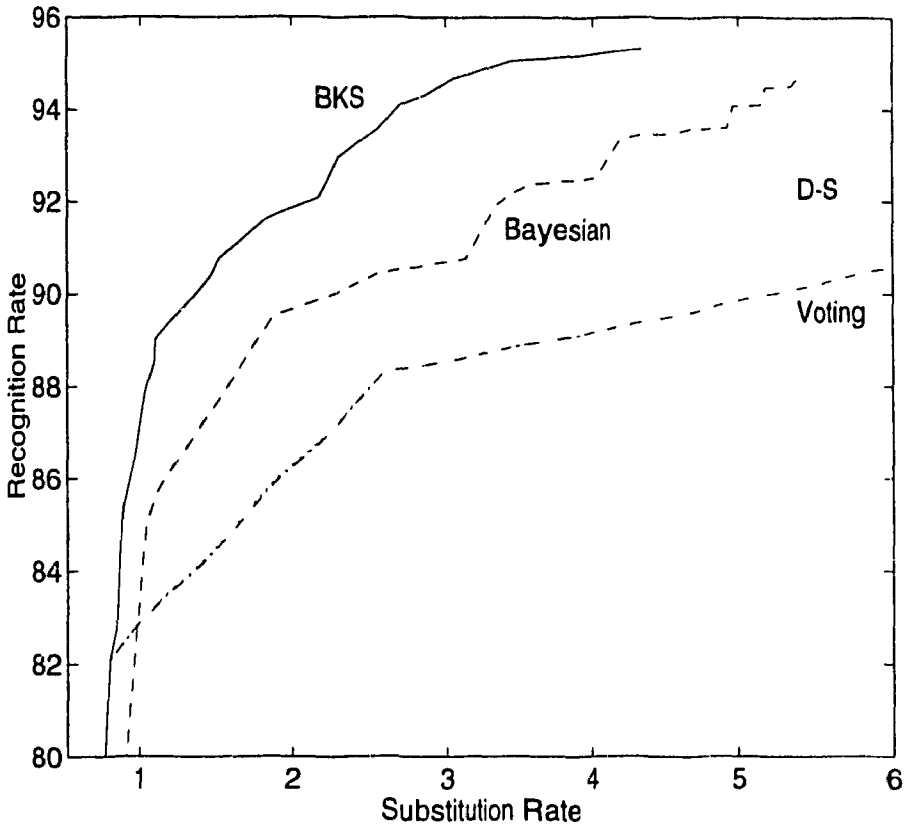


Figure 16: Graphic representation of the performances of four CME methods: e_1', e_1, e_2 and e_3; here, e_1' and e_1 have exactly the same recognition behavior.

classifiers	Rec.	Sub.	Rej.	Rel.
c_1	90.37	9.63	0.00	0.9037
c_2	90.93	9.07	0.00	0.9093
c_3	92.14	7.86	0.00	0.9214
$c_1 + c_2$	94.08	5.92	0.00	0.9408
$c_1 + c_3$	94.18	5.82	0.00	0.9418
$c_2 + c_3$	94.34	5.64	0.01	0.9434
$c_1 + c_2 + c_3$	95.31	4.36	0.33	0.9563

Table 11: Recognition performances of different number of classifiers using the BKS method on 41,377 testing samples.

different combination functions, and expects to achieve even better classification performance than the CME from only individual classifiers. In this experiment, besides the three individual classifiers, two combination functions based on the voting principle and the Bayesian formula are taken as two new classifiers as well. This means that in total 5 classifiers are used in this experiment. With the four CME methods, Table 12 lists the performances of these 5 classifiers with different thresholds. Figure 17 shows the corresponding recognition performances by using CME and consecutive CME for all four methods, respectively; the dotted line stands for the performance of the original CME, and the solid line the performance of consecutive CME. It is easy to observe that consecutive CME does not improve the recognition performance as expected. The BKS method produces almost the same performance in the two experiments, but surprisingly the Bayesian combination function even produces less efficient recognition results.

α	Rec.	Sub.	Rej.	Rel.
0.200	93.74	4.26	2.00	0.9565
0.400	93.73	4.03	2.23	0.9587
0.600	82.09	0.77	17.14	0.9908
0.800	82.09	0.77	17.14	0.9908
1.000	82.09	0.77	17.14	0.9908

(a) The voting method

α	Rec.	Sub.	Rej.	Rel.
0.000	95.13	4.87	0.00	0.9513
0.500	95.11	4.80	0.09	0.9520
0.800	95.00	4.70	0.30	0.9528
0.900	94.81	4.59	0.60	0.9538
0.995	94.03	4.13	1.85	0.9580
0.999	93.36	3.43	3.21	0.9645
1.000	91.00	2.61	6.39	0.9721

(b) The Bayesian method

α	Rec.	Sub.	Rej.	Rel.
0.000	95.14	4.86	0.00	0.9514
0.400	95.00	4.81	0.19	0.9518
0.800	94.98	4.79	0.23	0.9520
0.850	94.23	4.40	1.37	0.9554
0.900	93.19	3.73	3.08	0.9615
0.950	92.66	3.45	3.89	0.9641
0.990	90.58	2.98	6.45	0.9682

(c) The Dempster-Shafer method

α	Rec.	Sub.	Rej.	Rel.
0.000	95.32	4.35	0.33	0.9564
0.100	94.69	3.05	2.26	0.9688
0.300	92.97	2.28	4.76	0.9761
0.500	90.42	1.44	8.14	0.9844
0.700	87.87	1.04	11.09	0.9883
0.900	82.74	0.82	16.43	0.9901
0.950	82.09	0.77	17.14	0.9908

(d) The BKS method

Table 12: Recognition performances of consecutive CME by 5 classifiers: e_1, e_2, e_3, voting, and Bayesian on 41,377 samples.

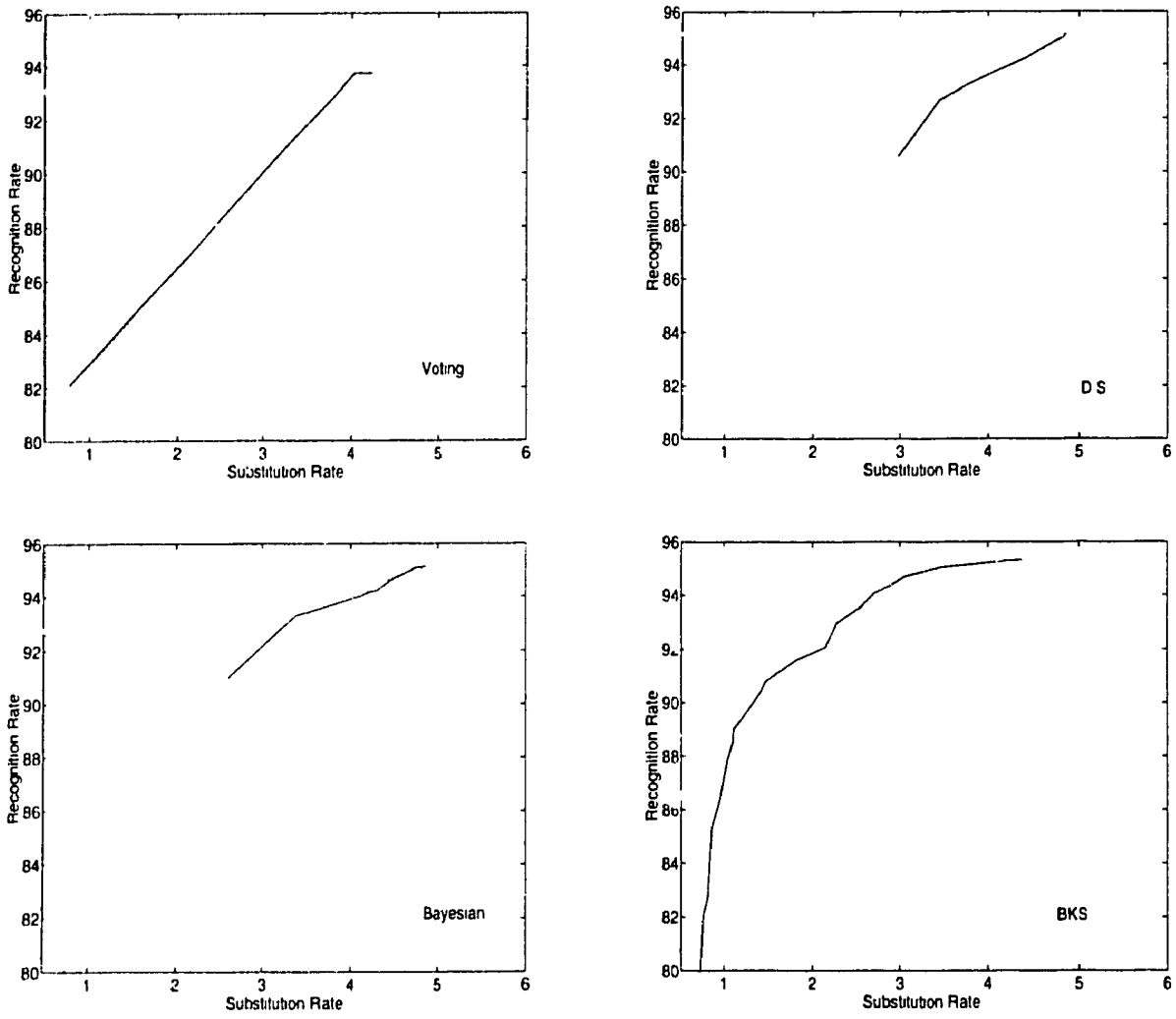


Figure 17: Recognition performance of consecutive CME, where for each combination method dotted lines stand for the performances of the original CME and solid lines for the performances of consecutive CME.

6.3.6 Analysis of Experimental Results

Through the above experiments, several observations can be drawn:

(1) All four CME methods perform much better than any individual classifier. This shows that by CME a recognition system with high recognition and low substitution rates is achievable.

(2) The BKS method modified by the second approach (*i.e.*, initializing the value of incoming samples of each class in each cell as 1 instead of 0) shows very promising recognition performance. In almost every situation, it outperforms the other three CME methods.

(3) The voting and Dempster-Shafer methods obtained similar performances in these experiments. This is mainly due to three conditions: first, no classifiers reject samples; second, the recognition rate of a classifier k is used for the corresponding *bpa* (basic probability assignment) as shown in [54] when this classifier offers its decision j_k; and third, all the three involved classifiers have similar performances (about 90% - 92% recognition rate). It was found that whenever these three conditions are satisfied, then the Dempster-Shafer method will become a voting-like method.

(4) All methods converge their performance curves near the point with a recognition rate of 82.09% and a substitution rate of 0.77%. This shows that whenever

all classifiers produce the same abstract-level classification output, there is no difference in the final output derived from different abstract-level CME methods. This also indicates whether the involved classifiers are able to achieve the required performance (*e.g.* in this case it is impossible to achieve a system performance with a required recognition rate of higher than 82.09%, and at the same time a substitution rate of less than 0.77%).

(5) The BKS method will not degenerate its performance even when the combined classifiers are strongly inter-dependent. However, the voting, Bayesian, and Dempster-Shafer methods may display quite biased performance because of dependence among classifiers.

(6) Heuristically, consecutive CME can improve recognition accuracy; however, it may also degenerate recognition accuracy. As for the BKS method, consecutive CME will not affects the overall recognition accuracy at all.

6.4 Experiments on Measurement-Level CME

Five measurement-level combination approaches are involved in this experiment, namely *Bodar Count, polynomial classifier, LCA, multi-layer perceptron* and *k-nearest-neighbor decision rule*. In abstract-level-CME experiments, a leave-one-out estimation is adopted. However, it is too expensive to use the same estimation strategy

to perform the experiment for measurement-level CME, because the knowledge derived from measurement values usually cannot be stored explicitly in matrix forms. Therefore, the 9 sets of testing numerals (41,377 samples) are further divided into two portions in this experiment for measurement-level CME. One is used for CME training, and the other for CME testing. Also, set 1 is used as a validation set, to prevent the over-training of CME. In the following experiments, sets 1-4 (19,353 samples) were chosen for training, and sets 5-9 (22,024 samples) for testing. Table 13 lists the classification accuracy of the three classifiers on the testing data set (22,024 samples).

The experiments serve two purposes: (1) to investigate the effectiveness of the proposed data transformation function T; and (2) to compare the CME performance of a multi-layer perceptron with those of other measurement-level CME approaches. In the following discussion, if a multi-layer perceptron adopts the proposed three improvement strategies discussed in Section 5.4, it is called a *modified* multi-layer perceptron; otherwise it is the original multi-layer perceptron or simply the multi-layer perceptron.

6.4.1 Experiment for Data Transformation

To investigate the effectiveness of the proposed data transformation function T, two other data transformation functions (M1 and M2) were implemented, and their transformed data were trained and tested by a three-layer perceptron (with 30 input

	Rec.	Sub.	Rel.
e_1	89.77	10.23	0.8977
e_2	90.50	9.50	0.9050
e_3	91.68	8.32	0.9168

Table 13: Recognition performances of individual classifiers using 22,024 testing samples (sets 5 - 9).

nodes, 20 hidden nodes, and 10 output nodes). It is believed that the best data transformation function is the one with the best recognition accuracy. These data transformation functions are: (suppose m_k^i belongs to the first kind of measurement)

T: *the proposed method*

$$t_k^i = \frac{\frac{1}{(m_k^i)^r}}{S_k}$$

and

$$S_k = \sum_{i=1}^{M} \frac{1}{(m_k^i)^r}.$$

M1: *division by the second preferred value* –

$$t_k^i = \frac{m_k^i}{A} \tag{42}$$

where A is the second smallest value among all measurements produced by classifier k on each sample. This function is adopted in [48].

M2: *division by statistical properties* –

$$t_k^i = \frac{m_k^i - u_k}{\sigma_k} \tag{43}$$

where $u_k = \frac{1}{M}\sum_{i=1}^{M} m_k^i$ and $\sigma_k^2 = \frac{1}{M}\sum_{i=1}^{M}(m_k^i - u_k)^2$.

Besides the three kinds of transformed measurements (M1, M2, and the proposed T), the original measurements are also taken into consideration in our experiments, so that the effect of data transformation can be revealed. However, there is a variable r in the proposed data transformation function T. In this experiment, r will be set as 1, 2 and 3, respectively. Table 14 lists the recognition results with respect to different data transformation functions and different values of r. It shows that a three-layer perceptron (1) without data transformation is unable to converge and perform well (16.96% recognition rate), (2) with improper data transformation (such as M1 and M2), it is still unable to achieve a good recognition result, and (3) with the proposed data transformation T, it can achieve highly improved performance (such as 96.20%, 97.05%, or 96.83% for the recognition rate). This experiment fully reflects not only the importance of data transformation for CME, but also the effectiveness of the proposed data transformation function. Since the highest recognition accuracy (97.05% recognition rate) is achieved when $r = 2$, the value of r will be set equal to 2 in the following experiments.

6.4.2 Experiment for Various CME Approaches

In addition to the original and modified multi-layer perceptrons (both have the same network topology with 30 input, 20 hidden, and 10 output nodes), four other CME approaches were developed for comparing the combination efficiency on the

	Rec.	Sub.	Rel.
ORI	16.96	83.04	0.1696
M1	55.67	44.33	0.5567
M2	68.34	31.66	0.6834
PRO-1	96.20	3.80	0.9620
PRO-2	97.05	2.95	0.9705
PRO-3	96.83	3.17	0.9683

Table 14: Recognition performances of different data transformation approaches with no rejection, where ORI stands for the original measurement values and PRO-n stands for the data processed by the proposed data transformation function T in Sec. 5.3.3, with $r = n$ and $n \in \{1, 2, 3\}$.

same transformed data (by using the proposed data transformation function T with $r = 2$). They are (1) Bodar Count [73], (2) polynomial classification [89], (3) Linear Confidence Aggregation [101], and (4) the k-Nearest Neighbor Decision Rule (k-NNR). Although there is no previous experiment based on k-NNR for CME, it is used in this experiment because CME has been considered as a pattern recognition problem, and k-NNR is one of the most commonly used pattern recognition functions. Simply speaking, k-NNR assigns an input pattern x to the class which has the most votes among the k nearest neighbors of x. In general, the larger the value of k is, the better the recognition accuracy will be. However, the computation time will become too long for practical use. In this experiment, the value of k is 9. With no rejection, Table 15 lists their recognition results on the testing data set. The last row of Table 15 is the performance of e_3 alone, the best performance among the

three individual classifiers to show the improvement obtained by CME. Obviously, this experiment shows that all six methods have achieved much better performance than any individual classifier; among them, the modified multi-layer perceptron has the best recognition accuracy (97.66%). Amazingly, in this experiment both the original multi-layer perceptron and the polynomial classifier obtain the same recognition accuracy (97.05%). Although the polynomial classifier has a recognition accuracy comparable to the original multi-layer perceptron, it generally takes more time to recognize a pattern. For example, in this experiment there are, in total, 465 inputs for the polynomial classifier, which contain 30 first-order and 435 second-order input features. To recognize an unseen pattern, it will take 4650 operations of both multiplication and addition. But the original multi-layer perceptron will take only 930 (*i.e.* $31 * 20 + 21 * 10$) operations of multiplication and addition. Therefore, concerning both speed and recognition accuracy, the modified multi-layer perceptrons indeed outperform other CME approaches. With threshold α of 0.05, Table 16 lists the confusion matrix of the modified three-layer perceptron, and Figure 18 shows a graphic representation of the different recognition performances with respect to different thresholds.

6.4.3 Analysis of Experimental Results

Some observations can be drawn from the above experiments in Sections 6.4.1 and 6.4.2:

CME Approach	Rec.	Sub.	Rel.
Bodar	94.36	5.64	0.9436
LCA	94.86	5.14	0.9486
9-NNR	96.54	3.46	0.9654
Poly	97.05	2.95	0.9705
Multi-Layer Perceptron	97.05	2.95	0.9705
Modified MLP	97.66	2.34	0.9766
e_3	92.15	7.85	0.9215

Table 15: Recognition performances of different CME approaches, where Poly stands for the polynomial classifier and Modified MLP for the multi-layer perceptron modified by the three strategies mentioned in sub-section 5.4.

	0	1	2	3	4	5	6	7	8	9	reject
0	2025	0	5	0	2	2	5	0	4	2	4
1	0	2709	1	2	1	2	1	1	0	1	5
2	2	5	2642	32	6	4	11	13	4	1	21
3	0	4	29	2404	3	7	1	8	3	3	11
4	8	1	1	0	2060	6	8	8	2	6	14
5	0	1	0	3	1	1969	10	2	3	0	7
6	12	1	0	0	0	5	1933	0	6	0	7
7	0	3	12	26	0	1	0	2073	0	51	20
8	7	0	6	2	2	4	4	0	1810	14	6
9	2	3	0	5	20	1	0	19	15	1847	11

Table 16: Confusion Matrix on the ITRI's numeral database, where the columns represent the recognized class labels and the rows represent the ground-truth class labels.

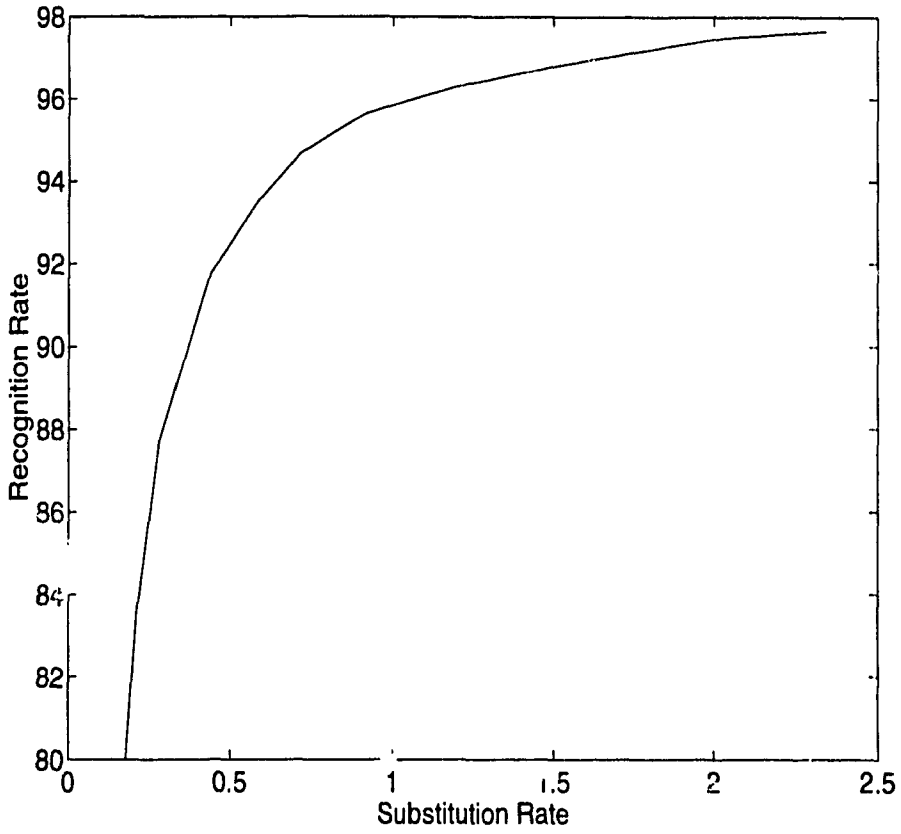


Figure 18: Graphic representation of the recognition performances of the modified three-layer perceptron.

(1) Data transformation plays a very important role in CME. For example, in our experiments, with a three-layer perceptron, only 16.96% of testing samples based on the original measurement values can be correctly recognized, but by using the proposed data transformation function a 97.66% recognition rate can be achieved.

(2) In CME, individual classifiers can be regarded as "classifiers" or "feature extractors", and from the viewpoint of classifier, some combination approaches are developed, such as the Bodar Count and LCA. From the viewpoint of feature extractor, generic pattern classification functions can be applied as combination functions, such as polynomial classifier, k-NNR, and multi-layer perceptron. From Table 15, obviously, the combination functions derived from the viewpoint of feature extractor generally produce better recognition performance than those from the viewpoint of classifier.

(3) When testing samples are not used for training, the best recognition accuracy for abstract-level CME is less than 96% (see Table 9), and that for measurement-level CME is higher than 97% (see Table 15). This reveals that measurement-level information is quite important for measurement-level CME.

(4) The three improvement strategies discussed in Section 5.4 for multi-layer perceptrons are responsible for increasing the recognition rate from 97.05% to 97.66%.

This enables us to conclude that multi-layer perceptrons are the most appro proate combination function for measurement-level CME.

Chapter 7

New Classifiers Based on CME Techniques

7.1 Introduction

So far, CME combines classifiers using different features or classification methodologies. In fact, the concept of CME can also be applied to design new classifiers by combining several classifiers, each of which makes use of subsets of a large set of features with a same classification function. For example, high-dimensional feature vectors can be divided into several vectors with lower dimensions, each of which is then used as an input to design its own classifier. In this way several classifiers can be constructed, and accordingly CME techniques are required to integrate the classification results of all these classifiers. The main reason for avoiding the use

of a single classifier with high-dimensional feature vectors is that high-dimensional feature vectors do not only increase computation complexity, but also produce implementation and accuracy problems [54]. Often, this phenomenon is called "the curse of dimensionality" [63]. For another example, an M-class recognition problem can be converted into $\frac{M(M+1)}{2}$ two-class recognition sub-problems. The objective of each recognition sub-problem is to have a high discrimination performance on its paired classes. Naturally, it requires the construction of a classifier for each pair of classes, called pair classifiers. Once this is done, CME techniques can then be applied to combine together the classification outputs of pair classifiers. In this chapter, we describe two experiments using CME methodologies to design the classifiers.

Section 7.2 describes the first experiment, "*Recognition by Parts*". Basically, it contains three steps. First, gradient features are extracted from each character image to form a feature vector. Second, a feature vector is divided into six sub-feature vectors. Each sub-feature vector indeed corresponds to the gradient features of a partial image, and will be used to construct a corresponding classifier which generates the measurement-level output information. Third, a three-layer perceptron is applied to aggregate the measurement-level output information of the six constructed classifiers, and it produces the final classification decision. Section 7.3 describes the second experiment, "*Recognition by Pair Classifiers*". First, an image algebraic feature extraction method is applied to each pair of classes to extract image features.

Then, a nearest neighbor classifier with a small number of prototypes is designed for each pair of classes, based on the algebraic features of training samples. Finally, a multi-layer perceptron is used to combine the measurement values of paired classes.

7.2 Recognition by Parts

Suen *et al.* [114, 115] have conducted several experiments to compare human and machine recognition capability by observing only subparts of the whole character image. This approach offers a better understanding into the relevant parts of characters for recognition. However, these experiments are performed without combination. The objective of this experiment is to further explore the concept of recognition by parts, and to design a new type of classifiers with the CME technique. Gradient features and statistics-based classification function are used in this experiment. A similar idea can be seen in the experiment performed by Franke [89].

7.2.1 Gradient Feature Extraction

It has been shown that gradient features which represent the magnitudes of character strokes in different directions are effective in character recognition [116]. Four directions are used in this experiment: $0^0, 45^0, 90^0$, and 135^0. Accordingly, four gradient operators (G_0, G_{45}, G_{90}, and G_{135}) are constructed to compute gradient features in the four directions respectively. Each gradient operator is represented by a $3 * 3$

mask G as

$$G = \begin{array}{|c|c|c|}\hline p_4 & p_3 & p_2 \\ \hline p_5 & p_0 & p_1 \\ \hline p_6 & p_7 & p_8 \\ \hline \end{array}$$

where $p_i, i = 1, \cdots, 8$, is a component of the mask in real number. In our experiment, the coefficients of the four gradient operators are designed to be

$$G_0 = \begin{array}{|c|c|c|}\hline 1 & \sqrt{2} & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -\sqrt{2} & -1 \\ \hline \end{array} \qquad G_{135} = \begin{array}{|c|c|c|}\hline \sqrt{2} & 1 & 0 \\ \hline 1 & 0 & -1 \\ \hline 0 & -1 & -\sqrt{2} \\ \hline \end{array}$$

$$G_{90} = \begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline \sqrt{2} & 0 & -\sqrt{2} \\ \hline 1 & 0 & -1 \\ \hline \end{array} \qquad G_{135} = \begin{array}{|c|c|c|}\hline 0 & -1 & -\sqrt{2} \\ \hline 1 & 0 & -1 \\ \hline \sqrt{2} & 1 & 0 \\ \hline \end{array}$$

Let (1) A_j^i, which stands for the jth image of class i, be a matrix in R^{r*n}; (2) $A_j^i(p,q)$ be the value of pixel (p,q) of A_j^i; and (3) $B_{j,0}^i(p,q)$, $B_{j,45}^i(p,q)$, $B_{j,90}^i(p,q)$, and

$B^t_{j,135}(p,q)$ be the four corresponding gradient features of $A^t_j(p,q)$; then

$$\begin{cases} B^t_{j,0}(p,q) &= A^t_j(p,q) \odot G_0 \\ B^t_{j,45}(p,q) &= A^t_j(p,q) \odot G_{45} \\ B^t_{j,90}(p,q) &= A^t_j(p,q) \odot G_{90} \\ B^t_{j,135}(p,q) &= A^t_j(p,q) \odot G_{135}. \end{cases} \tag{4.4}$$

Here, $\odot$ is a convolution operator with the following operation

$$\begin{aligned} A^t_j(p,q) \odot G = \quad & A^t_j(p,q) * p_0 + A^t_j(p+1,q) * p_1 + A^t_j(p-1,q+1) * p_2 \\ + \quad & A^t_j(p,q-1) * p_3 + A^t_j(p-1,q-1) * p_4 + A^t_j(p-1,q) * p_5 \\ + \quad & A^t_j(p-1,q+1) * p_6 + A^t_j(p,q+1) * p_7 + A^t_j(p+1,q+1) * p_8. \end{aligned}$$

7.2.2 Normalization of Gradient Features

Let $\mathcal{N}$ denote a linear operator[1] which normalizes an arbitrary size of image into a standard $p * q$ matrix. By using the same normalization operator with respect to the corresponding image, each gradient matrix is then normalized into a standard $p * q$ matrix as well. Suppose $Q^t_{j,0}(r,s), Q^t_{j,45}(r,s), Q^t_{j,90}(r,s), Q^t_{j,135}(r,s)$ are the four gradient features of component (r,s) of the normalized image of A^t_j, and this component (r,s) covers the pixels from the a_1th to a_2th rows and the b_1th to b_2th columns

[1]Usually, a nonlinear normalization produces better normalization result than a linear one. But, a linear normalization is used here mainly because of the simplicity of its mathematic treatment

of the original image; then

$$\begin{cases} Q^t_{j,0}(r,s) &= \frac{1}{(a_2-a_1+1)(b_2-b_1+1)} \sum_{x=a_1}^{a_2} \sum_{y=b_1}^{b_2} |B^t_{j,0}(x,y)|. \\ Q^t_{j,45}(r,s) &= \frac{1}{(a_2-a_1+1)(b_2-b_1+1)} \sum_{x=a_1}^{a_2} \sum_{y=b_1}^{b_2} |B^t_{j,45}(x,y)|. \\ Q^t_{j,90}(r,s) &= \frac{1}{(a_2-a_1+1)(b_2-b_1+1)} \sum_{x=a_1}^{a_2} \sum_{y=b_1}^{b_2} |B^t_{j,90}(x,y)|. \\ Q^t_{j,135}(r,s) &= \frac{1}{(a_2-a_1+1)(b_2-b_1+1)} \sum_{x=a_1}^{a_2} \sum_{y=b_1}^{b_2} |B^t_{j,135}(x,y)|. \end{cases} \tag{45}$$

In our experiment, both p and q are set to be 8.

7.2.3 Construction of Subpart Classifiers

Instead of using $8 * 8 * 4 = 256$ dimensions of features to construct a single classifier, the 256-dimension gradient features are further divided into six parts as shown in Figure 19, where the shaded areas in each standard matrix are not used for recognition, and the corresponding complete image is shown in Figure 20. The 6 parts are [1:3, 1:8], [4:5, 1:8], [6:8, 1:8], [1:8, 1:3], [1:8, 4:5], and [1:8, 6:8], where $[a_1 : a_2,\ b_1 : b_2]$ denotes the subpart contents from the a_1th to a_2th rows and the b_1th to b_2th columns of a standard $8 * 8$ matrix.

Based on training samples, each subpart constitutes a classifier by calculating the distances between the gradient features of this subpart of an input pattern and the corresponding average gradient features of this subpart of each class. Consequently, there are in total six classifiers. Suppose $m^t_0(r,s), m^t_{45}(r,s), m^t_{90}(r,s)$ and $m^t_{135}(r,s)$ are the four average gradient values of the component (a,b) of the $8 * 8$ standard

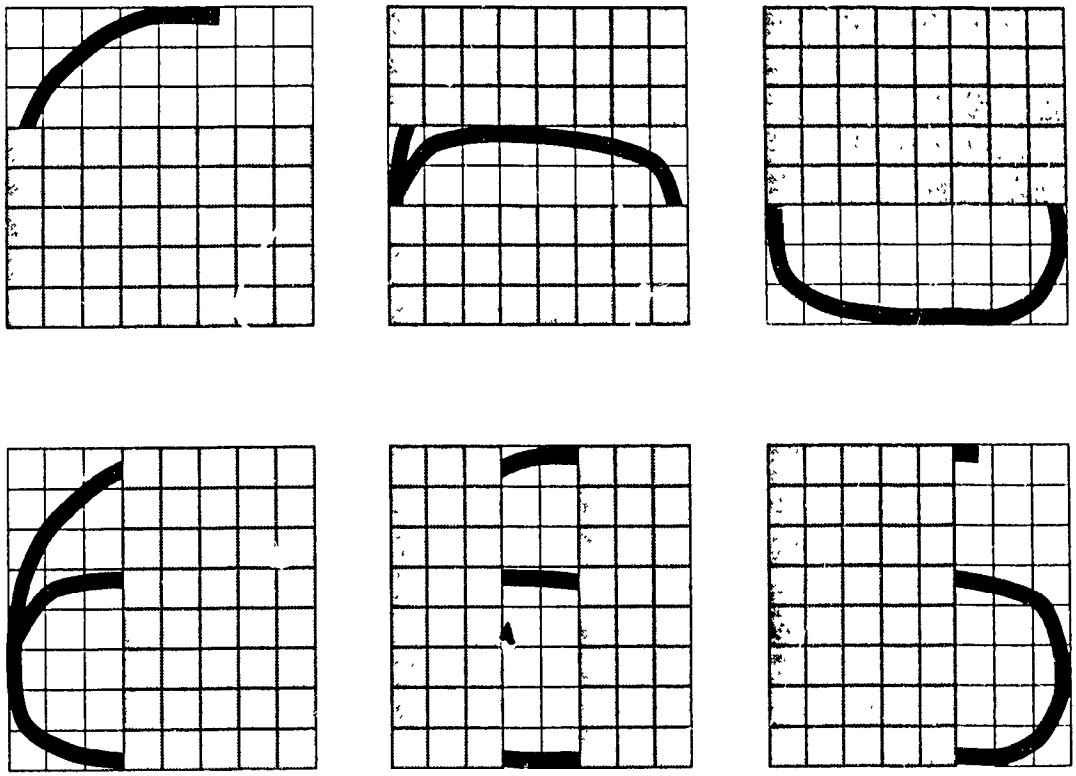


Figure 19: 6 parts of $8 * 8$ gradient features.

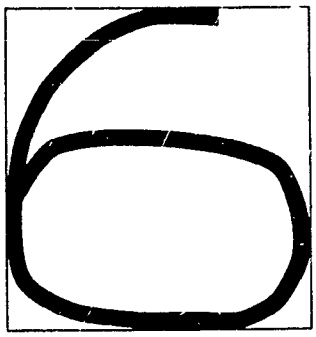


Figure 20: The complete image of numeral 6.

matix, respectively: then

$$m_0^i(r,s) = \frac{1}{N_i}\sum_{j=1}^{N_i} Q_{j,0}^i(r,s),$$

$$m_{45}^i(r,s) = \frac{1}{N_i}\sum_{j=1}^{N_i} Q_{j,45}^i(r,s),$$

$$m_{90}^i(r,s) = \frac{1}{N_i}\sum_{j=1}^{N_i} Q_{j,90}^i(r,s),$$

$$m_{135}^i(r,s) = \frac{1}{N_i}\sum_{j=1}^{N_i} Q_{j,135}^i(r,s),$$

where N_i is the total number of the training patterns of class i.

Let $R_0(r,s), R_{45}(r,s), R_{90}(r,s)$ and $R_{135}(r,s)$ be the four normalized gradient values of component (r,s) of an arbitrary image x. To subpart $[a_1 : a_2,\ b_1 : b_2]$, the distance $d^i(a_1,a_2,b_1,b_2)$ between the input pattern and class i is calculated by

$$\begin{aligned} d^i(a_1,a_2,b_1,b_2) = \sum_{r=a_1}^{a_2}\sum_{s=b_1}^{b_2}\Big(& \tfrac{1}{v_0^i(r,s)}(R_0(r,s)-m_0^i(r,s))^2 + \\ & \tfrac{1}{v_{45}^i(r,s)}(R_{45}(r,s)-m_{45}^i(r,s))^2 + \\ & \tfrac{1}{v_{90}^i(r,s)}(R_{90}(r,s)-m_{90}^i(r,s))^2 + \\ & \tfrac{1}{v_{135}^i(r,s)}(R_{135}(r,s)-m_{135}^i(r,s))^2 \Big), \end{aligned} \tag{46}$$

where $v_0^i(r,s), v_{45}^i(r,s), v_{90}^i(r,s)$ and $v_{135}^i(r,s)$ are the variances of $Q_{j,0}^i(r,s)$, $Q_{j,45}^i(r,s)$, $Q_{j,90}^i(r,s)$ and $Q_{j,135}^i(r,s)$ for all $j \le N_i$, respectively, *i.e.*,

$$v_0^i(r,s) = \frac{1}{N_i}\sum_{j=1}^{N_i}(Q_{j,0}^i(r,s)-m_0^i(r,s))^2,$$

$$v_{45}^i(r,s) = \frac{1}{N_i}\sum_{j=1}^{N_i}(Q_{j,45}^i(r,s)-m_{45}^i(r,s))^2,$$

$$v^t_{90}(r,s) = \frac{1}{N_t}\sum_{j=1}^{N_t}(Q^t_{j,90}(r,s) - m^t_{90}(r,s))^2,$$

$$v^t_{135}(r,s) = \frac{1}{N_t}\sum_{j=1}^{N_t}(Q^t_{j,135}(r,s) - m^t_{135}(r,s))^2.$$

7.2.4 Results of Combining the Classifiers

Because each classifier produces measurement-level information, every pattern has 60 (*i.e.* $6 * 10$) dimensions of output information, which are fed to a three layer perceptron. The number of hidden nodes is 20, so the perceptron is a 60-20-10 node architecture. The data used in the experiment come from the ITRI's numeral database. With the proposed data transformation function described in Section 5.3.2, the 60 dimensions of the transformed output features on Sets 1-4 are used as training data. In the testing stage, the following operations are performed on each sample:

1. Calculate the gradient features of the original image by Equation (44);
2. Normalize the gradient features by Equation (45);
3. With the normalized gradient features and the six constructed subpart classifiers, calculate the measurement-level outputs by Equation (46);
4. Input all the measurement outputs to the trained perceptron, and derive the final classification decision.

Sets 5-9 are used for testing and Table 17 lists the recognition results of the six classifiers on them respectively. With the three-layer perceptron, the final recognition rate

part	Rec.	Sub.
1	63.37	36.63
2	66.49	33.51
3	73.70	26.30
4	70.46	29.54
5	72.66	27.34
6	71.28	28.72

Table 17: Recognition performances of six individual sub-part classifiers with no rejection.

is 96.03% with no rejection. Table 18 shows the corresponding recognition distribution with respect to different thresholds. For comparison, the original 256 ($8 * 8 * 4$) gradient features per sample are also used to train a three-layer perceptron with 256, 30 and 10 nodes for the input, hidden and output layers respectively. Table 19 displays its recognition results. Although the size of the 60-20-10 net is much smaller than that of the 265-30-10 net, recognition by parts with 6 classifiers achieves better recognition performance than recognition by a single classifier using the original high-dimension gradient features.

7.3 Recognition by Pair Classifiers

In the following, a method based on algebraic feature extraction and classifier combining techniques is proposed to classify the images of M classes. First, an algebraic feature extraction method [117] is applied to $M(M-1)/2$ pairs of classes to

λ	Rec.	Sub.	Rej.	Rel.
0.0	96.03	3.97	0.00	0.9603
0.2	94.13	2.35	3.52	0.9757
0.3	91.37	1.23	7.41	0.9868
0.6	86.90	0.65	12.44	0.9924
0.9	80.74	0.36	18.90	0.9955

Table 18: Recognition performances of combining six sub-part classifiers.

λ	Rec.	Sub.	Rej.	Rel.
0.0	95.26	4.74	0.00	0.9526
0.1	94.34	3.48	2.18	0.9644
0.3	93.80	3.10	3.10	0.9680
0.5	92.33	2.31	5.35	0.9755
0.7	90.51	1.79	7.70	0.9801
0.8	81.57	1.17	17.27	0.9859

Table 19: Recognition performances by a trained 256-30-10 perceptron which uses the 256-dimension gradient features as input.

extract the image features. Then, a nearest neighbor classifier with a small number of prototypes is designed for each pair of classes, based on the extracted algebraic features of training samples. Finally, a neural network is trained to combine the measurement values of an input image with respect to each of the $M(M-1)/2$ classifiers. More details are described in our research work [118, 119].

7.3.1 Extraction of Algebraic Features from Character Images

Algebraic feature extraction for image recognition has attracted much attention in recent years. In many pattern recognition applications, such as human face recognition [120], character recognition [121, 122], *etc.*, the objects to be recognized are usually described by normalized images. In such cases, algebraic feature extraction methods have been proven to be very useful. Algebraic features are extracted from various algebraic matrix transforms or decompositions. The main characteristic of algebraic features is that they represent the intrinsic attributions of an image. In a recent paper [117], the idea of using *optimal discriminant criteria* to extract algebraic features of images was proposed, and an algebraic feature extraction technique was developed based on a generalized Fisher optimal criterion. In the following, a brief review of the method is presented first.

Let a training image be denoted by A_j^i, where $i = 1, 2, \cdots, M$ indicates the image class. M is the total number of classes, $j = 1, 2, \cdots, N_i$, and N_i the total number of training images in the ith class C_i. A_j^i is a matrix in $R^{r \times n}$. The idea of the method

is to find a discriminant projection vector u in R^n, so that the set of vectors $A_j^i u$, $i = 1, 2, \cdots, K$, has the *minimum within-class* and *maximum between-class scatters*. Suppose the mean images of the ith class of training images and of the K classes are represented by A^i and $\bar{A}$, respectively, and defined by the formulae:

$$A^i = \frac{1}{N_i} \sum_{j=1}^{N_i} A_j^i \tag{17}$$

and,

$$\bar{A} = \sum_{i=1}^{K} P_i A^i \tag{18}$$

where P_i, $i = 1, 2, \ldots, M$, is *a priori* probability of the ith class. Then, the generalized between-class scatter matrix D_b, within-class scatter matrix D_w, and Fisher criterion function $J(x)$ can be calculated by Equations (49 - 51):

$$D_b = \sum_{i=1}^{M} P_i (A^i - \bar{A})^T (A^i - \bar{A}) \tag{49}$$

$$D_w = \sum_{i=1}^{M} P_i \frac{1}{N_i} \sum_{j=1}^{N_i} (A_j^i - A^i)^T (A_j^i - A^i) \tag{50}$$

and,

$$J(x) = \frac{x' D_b x}{x' D_w x} \tag{51}$$

where x is a vector in R^n.

Suppose r $(2 \leq r \leq n)$ and $U = \{u_i\}_{i=1}^{r}$ are the number and the set of the optimal discriminant projection vectors, respectively. Then, each vector of U can be calculated step by step using the following procedure:

(1) u_1 is the unit vector maximizing the generalized Fisher criterion function $J(x)$ in R^n.

(2) The ith optimal discriminant projection vector u_i $(2 \leq i \leq r)$ is determined by solving the following problem:

$$\max\left(J(x)\right)$$
$$u_j' u_i = 0, \qquad j = 1, ..., i-1.$$
$$|u_i| = 1$$

Details about the physical meaning of the generalized Fisher criterion function, the optimal projection vectors, and the algorithm of solving u_i can be found in [117].

After the set $U=\{u_i\}_{i=1}^{r}$ of optimal discriminant projection vectors has been determined, based on the above method and training image samples, the algebraic features $Y_1, ..., Y_r$ of an input image can be extracted by the formula:

$$Y_i = Au_i, \ i = 1, \cdots, r. \tag{52}$$

7.3.2 Calculation of the Prototypes of Pair-Class

For each pair of classes i and j $(i < j)$, their corresponding prototypes can be calculated according to the following steps:

(1) Calculate the optimal discriminant projection vecters $u_l^{(ij)}$, $1 \leq l \leq r$, based on the training samples of classes i and j;

(2) Calculate the algebraic features of all training samples by projecting them onto the optimal discriminant projection vectors according to the formula:

$$Y_l^{(ij)} = Au_l^{(ij)},\ l = 1, \cdots, r. \tag{53}$$

where A is an image sample belonging to class i or class j;

(3) For each image sample, constitute a feature vector $X^{(ij)}$ from its algebraic features $Y_l^{(ij)}$, $l = 1, \cdots, r$, as follows:

$$X^{(ij)} = (X_1, X_2, ..., X_r)^T \tag{54}$$

where $X_l^T = Y_l^{(ij)}$, $l = 1, \cdots, r$;

(4) For the feature vector sets, calculate the prototypes $R_1^{(ij)}$, $R_2^{(ij)}, \cdots, R_p^{(ij)}$ of class i and the prototypes $R_{p+1}^{(ij)}$, $R_{p+2}^{(ij)}, \cdots, R_{2*p}^{(ij)}$ of class j, where p is a given integer. Several methods such as k-means method [13] or neural network-based prototype optimization methods [23, 123, 124] can be used to calculate the prototypes. Based on the prototypes, a nearest neighbor classifier can be built for the pair classes i and j. Only a small positive integer p is required to calculate the distance measurement values.

7.3.3 Calculation of Distance Measurement Values of Images

For an arbitrary image sample A of M classes, its distance measurement values can be calculated based on the prototypes and the above calculated optimal discriminant projection vectors, using the following procedure:

(1) Calculate the algebraic features corresponding to the optimal discriminant projection vectors of each pair of classes i and j by Equation (53) and then constitute the feature vector $X^{(ij)}$ by Equation (54).

(2) Calculate the first k minimum distances $d_{i-1}^{(ij)}$, $d_{i-2}^{(ij)}$,$\cdots$, $d_{i-k}^{(ij)}$ between the feature vector $X^{(ij)}$ and the prototypes $R_1^{(ij)}$, $R_2^{(ij)}$,$\cdots$,$R_p^{(ij)}$, and the first k minimum distances $d_{j-1}^{(ij)}$, $d_{j-2}^{(ij)}$,$\cdots$, $d_{j-k}^{(ij)}$ between $X^{(ij)}$ and the prototypes $R_{p+1}^{(ij)}$, $R_{p+2}^{(ij)}$,$\cdots$,$R_{2*p}^{(ij)}$, respectively;

(3) Constitute a new feature vector X_m from the distance measurement values of image A related to each pair of classes:

$$X_m = (d_{1-1}^{(12)},\cdots,d_{1-k}^{(12)},d_{2-1}^{(12)},\cdots,d_{2-k}^{(12)},\cdots,d_{M-1}^{((M-1)M)},\cdots,d_{M-k}^{((M-1)M)})^T \quad (55)$$

The dimensionality of X_m is $2 * k * M(M-1)/2$.

2.4 Data Classification and Experimental Result

The feature vectors of distance measurement values will be used as features for image classification. Again, a three-layer perceptron with the Generalized Delta Rule is used to serve the classification task.

The input patterns are first size-normalized to 16 × 16 pixels. Then, six optimal discriminant projection vectors per pair of classes are calculated (*i.e.* $r = 6$), and 20 prototypes per class corresponding to the set of algebraic features of each pair of

classes, are derived by a k-means algorithm. The feature vectors of distance measurement values of training and testing samples are extracted based on these optimal discriminant projection vectors and prototypes, according to the method described in Section 3. Since the number of classes is 10, there are totally 45 pairs of classes. k is taken as 1 in this experiment; therefore, the dimensionality of the feature vectors of distance measurement values is 90. The 4,000 feature vectors are used to train a three-layer perceptron with 90 input nodes, 90 hidden nodes and 10 output nodes by the standard back propagation algorithm. After training, the net is used as the classifier to recognize the feature vectors of the testing set. For Sets 5-9 of the ITRI's numeral database, the highest recognition rate achieved is 96.55%, with no rejection. More detailed results with rejection are given in Table 20. Compare with the results obtained from recognition by parts, the combination of pair classifiers produces a higher recognition accuracy. This indicates that the extracted algebraic features are effective in describing the distinguishing information between paired classes. However, the dimension of feature produced by pair classifiers is proportional to $O(M^2)$, but that by recognition by parts is only $O(M)$. Therefore, it is impractical to apply pair classifiers directly to problems with a large number M, *e.g.* Chinese character recognition. In fact, even when M is not very large such as alphanumerical recognition which contains 62 classes (*i.e.* 26 upper-case and 26 little-case alphabets, and

λ	Rec.	Sub.	Rej.	Rel.
0.0	96.55	3.45	0.00	0.9655
0.1	95.52	2.00	2.48	0.9795
0.2	94.96	1.60	3.44	0.9834
0.3	94.21	1.15	4.64	0.9879
0.4	93.18	0.94	5.88	0.9900
0.5	91.95	0.70	7.35	0.9924
0.6	90.31	0.50	9.19	0.9946
0.7	87.54	0.31	12.15	0.9965

Table 20: Recognition performances of "Recognition-by-Pair-Classifiers".

10 munerals), there are 1891 (61 * 31) pair classes. Because of the curse of dimensionality, it will require a very large number of training samples to achieve a good recognition performance; otherwise even when a trained perceptron performs well on training samples, it may perform poorly on unseen patterns.

Chapter 8

Prototype Optimization

As stated in Chapter 5, measurement-level CME turns out to be a generic pattern classification problem. Therefore, various pattern classification methods can be applied to combine the decisions of classifiers. Then, it is important for us to further improve the performance of pattern classification processes which can benefit CME in two aspects. First, it will dirtectly produce a better CME performance. Second, it can be used to construct better classifiers. In a multi-classfier recognition system, any improvement of the individual classifiers will correspondingly improve the performance of CME. In this thesis, we also made an effort to improve the effectiveness of the commonly-used nearest neighbor classification by deriving optimal prototypes.

8.1 Introduction

Nearest neighbor rule is well known due to its simplicity and high discriminant capability. It has been widely used especially in the cases of incomplete knowledge of the class probability densities for the entire pattern space. However, the implementation of a nearest neighbor classifier is computationally expensive in terms of storage space and computing time if the number of prototypes is large. Many attempts [21, 20, 125, 126, 127, 128, 129] in the past have been made to alleviate the computational burden of nearest neighbor classifiers. Generally speaking, traditional research on nearest-neighbor classifiers can be divided into two categories: *fast nearest-neighbor searching* and *prototype optimization*. In the first category, research works [21, 125, 126, 127] were focused on the use of fast algorithms to search for the nearest neighbor. For the second category, efforts [20, 23, 42, 128, 129, 81, 82] have been devoted to prototype optimization, *i.e.* to reduce the number of training samples as reference models on condition that the classification performance with a reduced set is at least as good as with a complete set. Our work presented in this chapter belongs to the above second category.

Recently, neural network methodology was extended to optimize the prototypes of nearest-neighbor classifiers [23, 42, 81, 82]. In [42, 81], Learning Vector Quantization(LVQ) neural network was used as clustering method to optimize prototypes of nearest neighbor classifiers. In Yan' work [23, 82], a set of prototypes are first selected

from the training samples to build a nearest neighbor classifier. The classifier is then mapped to a three-layer network of which each hidden node represents a prototype and the weights of connections between a hidden node and the input nodes are initially set to be equal to the feature values of the corresponding prototypes. Based on a gradient descent algorithm, the three-layer network is adaptively trained to minimize a defined error function and to derive optimized prototypes. After the training process, the three-layer network is mapped back to a nearest neighbor classifier with new and optimized prototypes.

The new method of optimizing prototypes for nearest neighbor classifiers proposed here is based on Yan's method [23, 82]. In Yan's method, the criteria for updating prototypes and for using the trained prototypes to build a nearest neighbor classifier are in fact inconsistent, which will be explicitly explained in Section 8.2. This drawback is eliminated in our newly modified model. In this new method, a novel network architecture and error function are designed to achieve consistent criteria for updating prototypes and for using the trained prototypes to build a nearest neighbor classifier. The relationship between the present method and LVQ2 is also revealed from a theoretical base. In Section 8.2, a brief review of Yan's method is introduced. The new prototype optimization method is then described in Section 8.3. Section 8.4 discusses the relation between LVQ2 and the present method. Finally, a comparison of the new method, Yan's method and LVQ2 are provided in Section 8.5, which shows

that the present method is consistently superior to the other two.

8.2 Yan's Prototype Optimization Method

Yan [82] used a three-layer network to produce prototypes for a nearest neighbor classifier. This net consists of input, hidden, and output layers. The total number of input nodes is N, which is equal to the dimension of the input features; and the total number of output nodes is K, which is equal to the total number of classes. In the hidden layer, each node corresponds to a prototype. In the output layer, node k contains the classification information of class k. Suppose there are J hidden nodes in total, $r_j = [r_{j1}, r_{j2}, \cdots, r_{jN}]^T$ is the prototype represented by hidden node j, and x is an input vector; then the input to hidden node j is

$$\begin{aligned} u_j &= d^2(x, r_j) \\ &= (x_1 - r_{j1})^2 + \cdots + (x_N - r_{jN})^2. \end{aligned} \tag{56}$$

The output of the hidden node is

$$y_j = \frac{2}{1 + e^{(u_j)}}. \tag{57}$$

The output of output node k is

$$z_k = \frac{1}{1 + e^{-(\sum_{j \in S_k} A_1 y_j + \sum_{j \notin S_k} A_2 y_j)}}, \tag{58}$$

where S_k is the index set in which each index corresponds to a hidden node that represents a prototype belonging to class k, A_1 is the connection weight from output

node k to each node of S_k, and A_2 is the connection weight from output node k to each hidden node not belonging to S_k.

The error function E is defined as

$$E = \frac{1}{2}\sum_{k=1}^{K}|z_{k0} - z_k|^2 \tag{59}$$

where z_{k0} and z_k are the desired and actual values of output node k. According to the generalized delta rule, r_{ji} is updated with the following increment

$$\triangle r_{ji} = \alpha \sum_{k=1}^{K}(z_{k0}-z_k)z_k(1-z_k)*[\sum_{j\in S_k} A_1 y_j(2-y_j)(x_i-r_{ji}) + \sum_{j\notin S_k} A_2 y_j(2-y_j)(x_i-r_{ji})] \tag{60}$$

where α is the learning rate. Although α in the original formula proposed by Yan is a constant, it is well known that in clustering analysis a variable learning rate in general will produce better clustering results than a constant learning rate. Therefore, α in our experiments has been extended to $\alpha(n)$, and Equation (61) is used to replace Equation (60):

$$\triangle r_{ji} = \alpha(n) \sum_{k=1}^{K}(z_{k0}-z_k)z_k(1-z_k)*[\sum_{j\in S_k} A_1 y_j(2-y_j)(x_i-r_{ji}) + \sum_{j\notin S_k} A_2 y_j(2-y_j)(x_i-r_{ji})] \tag{61}$$

where $\alpha(n)$ is monotonically decreased verse time, that is $\forall n\ 0 < \alpha(n+1) < \alpha(n)$, and n is the iteration number of the whole training data set. After training, the trained net is mapped back to a nearest neighbor classifier. For recognition, an input sample x is assigned to the class of prototype for which u_j is the minimum.

General speaking, the idea to use neural networks for optimizing pattern prototypes is innovative. But, from the above description we know that during the training process the prototypes are updated to reduce the global error E in which z_k $(1 \leq k \leq K)$ is computed from the distances between the input training sample and all the prototypes. However, when the trained net is mapped back to a nearest neighbor classifier, an input sample is assigned to the class of which the distance between a prototype and the input sample is minimum. In fact, the criteria for prototype updating (*i.e.* training) and for classification are inconsistent. From the point of view of a trained neural network, an input sample will be classified as class k_1 if $|z_{k_10} - z_{k_1}|$ is the minimum value of $|z_{k0} - z_k|$ $(1 \leq k \leq K)$. But, from the view point of a nearest neighbor classifier, the sample will be classified as class k_2 if the distance between a prototype of class k_2 and the input sample is minimum (*i.e.* $u_j = \min_{l \in \{1 \ldots J\}} u_l$ and $j \in S_{k_2}$). There is no guarantee that k_1 equals k_2 is always true. In practice, this inconsistency may considerably degrade the recognition accuracy of this method, which has been confirmed by our experiments.

8.3 A Novel Prototype Optimization Method

According to the above discussion, a new prototype optimization method [123, 124] is proposed, which improves on Yan's in two aspects: *network architecture* and *the definition of error function.*

8.3.1 Network Architecture and Node's Function

The new model contains a four-layer network as shown in Figure 21. These four layers are called the input, first- and second-hidden, and output layers, respectively. The number of nodes in the first three layers are the same as those used in Yan's model which are represented by N, J and K respectively, and there is only one node in the output layer. In our method, we use the nodes of the first-hidden layer to represent the prototypes of classes. Assume that the nodes in each layer are named to relate to that layer, *e.g.* the first-hidden nodes are the nodes of the first-hidden layer. Between the input and first-hidden layers, each first-hidden node is fully connected to the input nodes. However, between the first- and second-hidden layers, each second-hidden node k only connects to those first-hidden nodes which contain the prototypes of class k. The output node connects to all second-hidden nodes. Except the weights between the first-hidden and input layers, all connection weights of the network are equal to 1.

In order to design an appropriate error function, the output functions of the first- and second-hidden nodes are also changed accordingly. The newly defined node and error functions lead to the derivation of a simple but effective prototype update rule, which is described in Section 8.3.3. Instead of using a sigmoid function, each first-hidden node j uses one half of the input value as its output:

$$y_j = \frac{1}{2} d^2(x, r_j)$$

$$= \frac{1}{2}[(x_1 - r_{j1})^2 + \cdots + (x_N - r_{jN})^2]. \quad (62)$$

where r_j is the weight vector of prototype j. The output of a second-hidden node k is

$$z_k = \min_{j \in S_k} y_j \quad (63)$$

where S_k is the index set in which each index corresponds to a first-hidden node that represents a prototype belonging to class k. Equation (63) specifies that a second-hidden node k registers the index of the first-hidden node which belongs to class k and possesses the minimum value among the set $\{y_j | \ j \in S_k\}$, and to pass this value to the output node. The output node simply registers the index of the second-hidden node M, which has the minimum value among the set $\{z_k | k \in \{1 \cdots K\}\}$, *i.e.* $z_M = \min_{k=1}^{K} z_k$.

8.3.2 Error Function

Another major difference between the proposed method and Yan's is that a new error function is defined. Although the original error function in Equation (59) is commonly used to search the least-mean-square-error solution, this error function, in fact, does not directly correspond to the classification error by a nearest neighbor classifier[1]. Based on the node functions defined in Section 8.3.1, a new error function

[1] A nearest neighbor classifier increases the error by one whenever it makes a wrong classification, otherwise there is no increment of error at all.

E_{new} strongly correlated to the error of a nearest neighbor classifier is proposed as

$$E_{new} = 1 - e^{\frac{z_M - z_T}{A(n)}} \tag{64}$$

where M is the index of the class having the minimum distance to the input pattern x, T is the ground-truth class label of x, $A(n)$ is the *bandwidth* parameter, sometimes called the width of the *receptive field* of the sigmoid function, which governs the size of the active area of the sigmoid function, and $0 < A(n+1) < A(n)$ for all n in integer set. It is clear that according to definition z_M is always equal to or smaller than z_T. When z_M equals to z_T, it means that the trained network has made a correct classification. According to Equation (64), no increment of error will occur due to the current classification. When z_M is smaller than z_T, it indicates that the current pattern is mis-classified. Equation (64) then presents a certain increment of error. Here, $A(n)$ functions like a "*temperature*" parameter in the simulated annealing process [40], starting at a high value and going down. Interestingly, when $A(n)$ approaches 0, E_{new} becomes a binary function as

$$E_{new} = 0, \text{ if } z_M = z_T \ (i.e. \text{ correct classification}),$$

and

$$E_{new} = 1, \text{ if } z_M \neq z_T \ (i.e. \text{ wrong classification}).$$

This indicates that the newly defined error function directly corresponds to the classification error by the nearest neighbor classifier, especially when the temperature is low.

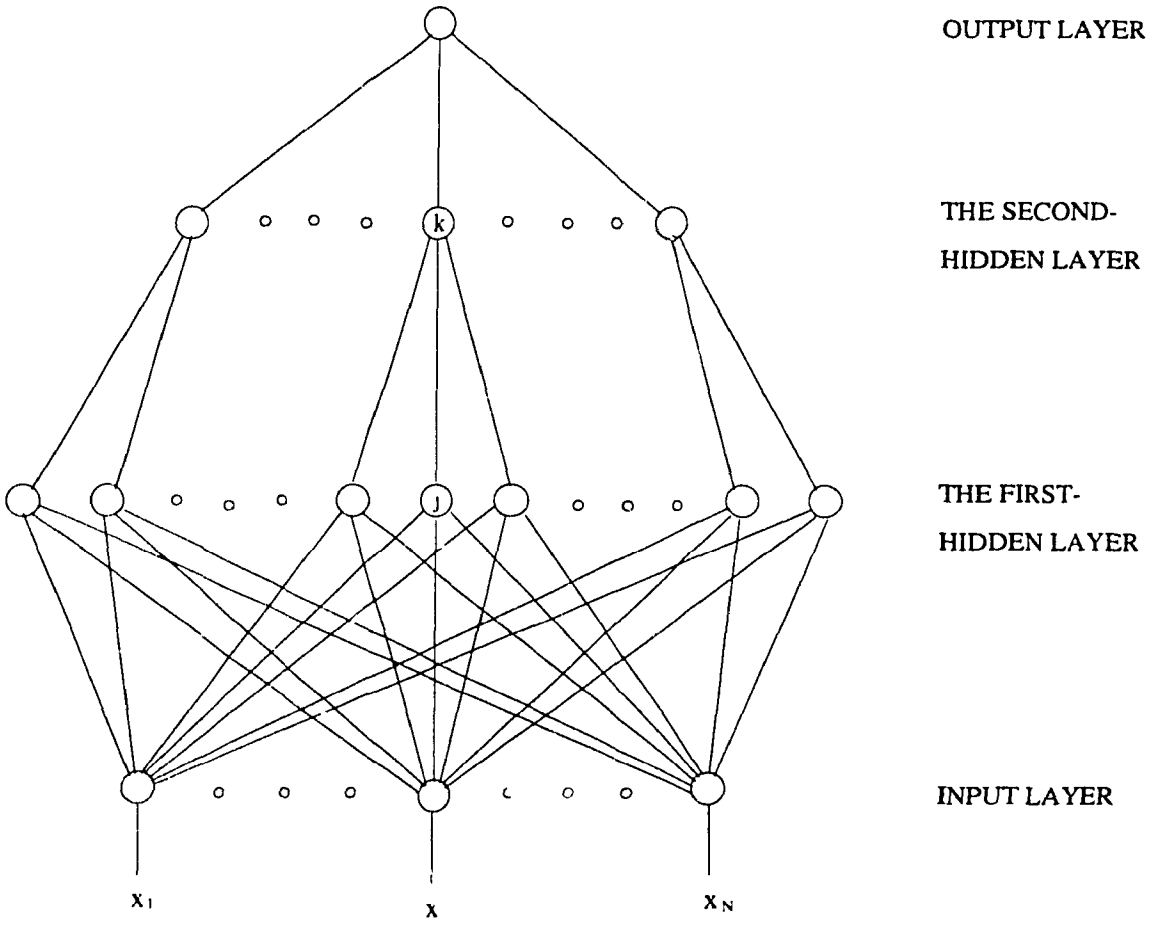


Figure 21: Diagram of the modified network.

8.3.3 The Prototype Update Rule

According to the generalized delta rule, r_{ji} can be updated as follows:

$$\triangle r_{ji} = -\alpha(n)\frac{\partial E_{new}}{\partial r_{ji}} \tag{65}$$

where $\alpha(n)$ is a monotonically-decreasing learning rate. Since E_{new} is expressed by output z_M and z_T, by using the chain rule, item $\frac{\partial E_{new}}{\partial r_{ji}}$ becomes

$$\frac{\partial E_{new}}{\partial r_{ji}} = \sum_{k=1}^{K} \frac{\partial E_{new}}{\partial z_k}\frac{\partial z_k}{\partial r_{ji}}. \tag{66}$$

From Equation (64), we obtain

$$\frac{\partial E_{new}}{\partial z_k} = \begin{cases} -\frac{1}{A(n)} * e^{\frac{z_M - z_T}{A(n)}} & \text{, when } z_M \neq z_T \text{ and } k = M; \\ \frac{1}{A(n)} * e^{\frac{z_M - z_T}{A(n)}} & \text{, when } z_M \neq z_T \text{ and } k = T; \\ 0 & \text{, otherwise.} \end{cases} \tag{67}$$

The calculation of $\frac{\partial z_k}{\partial r_{ji}}$ can also be expressed by the chain rule of two partial derivatives, that is

$$\frac{\partial z_k}{\partial r_{ji}} = \frac{\partial z_k}{\partial y_j}\frac{\partial y_j}{\partial r_{ji}}. \tag{68}$$

Interestingly, the first derivative becomes

$$\frac{\partial z_k}{\partial y_j} = \begin{cases} 1 & \text{, when } y_j = \max_{l \in S_k} y_l; \\ 0 & \text{, otherwise;} \end{cases} \tag{69}$$

and the second derivative is

$$\frac{\partial y_j}{\partial r_{ji}} = -(x_i - r_{ji}). \tag{70}$$

Therefore, putting Equations (65 - 70) together, we obtain

$$\triangle r_{ji} = \begin{cases} -\eta(n) e^{\frac{z_M - z_T}{A(n)}} (x_i - r_{ji}) & , \text{ when } z_M \neq z_T \text{ and } y_j = \max_{l \in S_M} y_l; \\ \eta(n) e^{\frac{z_M - z_T}{A(n)}} (x_i - r_{ji}) & , \text{ when } z_M \neq z_T \text{ and } y_j = \max_{l \in S_T} y_l; \\ 0 & , \text{ otherwise;} \end{cases} \quad (71)$$

where $\eta(n) = \frac{\alpha(n)}{A(n)}$. To ensure the convergence of the prototype optimization process, $\eta(n)$ should be a monotonically decreasing function.

8.3.4 Geometry Interpretation of the Prototype Update Rule

Equation (71) shows that: (1) if the network makes a correct classification (*i.e.* $z_M = z_T$), then no prototype will be updated; (2) if the network makes a wrong classification (*i.e.* $z_M < z_T$), two prototypes will be updated to reduce error (one is moved toward the current sample and the other is moved away from it); and (3) incremental weight $\triangle r_{ji}$ is influenced mostly by two items[2] $x_i - r_{ji}$ and $e^{\frac{z_M - z_T}{A(n)}}$. Undoubtedly, the first item satisfies our intuition that when the distance between x_i and r_{ji} is large, then $\triangle r_{ji}$ should be large as well. The second item $e^{\frac{z_M - z_T}{A(n)}}$ describes that within a complete iteration of the overall training data, a larger update takes place when z_M and z_T are close, and a smaller update when z_M is much smaller than z_T. In other words, if a network has made a wrong classification (*i.e.* $z_M \neq z_T$) on the identity of a pattern and it is far from making it right (*i.e.* $z_M \ll z_T$), then little is learned from this pattern. On the contrary, when the network is very close to

[2]For the sake of simplicity, the influence of $\eta(n)$ is not discussed here.

make the correct classification (*i.e.* $z_M \approx z_T$), then a large amount of learning will be required.

It is easy to verify that the class corresponding to the smallest value of z_k is the same as the one corresponding to the smallest u_j. This reveals that the present method and the nearest neighbor classifier actually have the same recognition performance, and enables us to classify an input pattern by using the nearest neighbor classifier with the prototypes produced by this method. In fact, consistency between the training and recognition stages is the primary reason that this modified model can outperform Yan's.

8.4 Relation Between LVQ2 and The Present Method

Learning Vector Quantization (LVQ) is a famous clustering method used for nearest neighbor classification. In this section, the relationship between LVQ-based prototype optimization methods and the present one is revealed. Due to our focus on supervised learning process, LVQ2 is the one used for comparison. When the Euclidean distance metric is used in LVQ2 for distance measurement, with the same notation and definition used in Section 8.3.1, LVQ2 can be expressed as follows: for a pattern x,

(1) When $z_M = z_T$, then no prototypes are updated;

(2) When $z_M \neq z_T$, pattern x is misclassified, then two steps will be performed:

$$\text{Reinforced Learning:} \quad \triangle r_{t_t} = \alpha(n) * (x_t - r_{t_t}), \qquad \text{when } y_t = \max_{j \in S_T} y_j;$$

$$\text{Anti-reinforced Learning:} \quad \triangle r_{m_t} = -\alpha(n) * (x_t - r_{m_t}), \quad \text{when } y_m = \max_{j \in S_M} y_j.$$

Comparing the prototype update rule of LVQ2 with that of the present method (see Equation (71)), if $A(n)$ is set to be a fixed large value (such as $A(n) = 10000.0$) and $\eta(n)$ equals $\alpha(n)$, then the two update rules become the same. This realization manifests that LVQ2 is a special case of the present method.

It is well known that although LVQ-based clustering methods are capable of reducing their corresponding error functions, they are liable to converge on local minima rather than global minima. This may result in quite different performances with respect to different prototype initializations. Two approaches are generally adopted to alleviate this problem. The first is to minimize an error function by using the global error optimization techniques, such as genetic algorithms [130, 131] or simulated annealing processes [40, 132, 133]. The second is to define an error function which changes with time. Hypothetically, a local minimum of an error function is unlikely to be a local minimum of another error function. Therefore, by continuously changing an error function, local minima may be avoided. In fact, the present method intrinsically takes both approaches into consideration. First, the present error function E_{new} is not fixed because it changes continuously according to parameter $A(n)$.

Second, the present error function E_{new} consists of a so called "deterministic annealing" property [132], which incorporates a certain degree of *randomness* movement and is deterministically optimized at each temperature sequentially, starting from a high temperature and going down.

8.5 Experiments

A series of experiments have been conducted to compare the proposed, LVQ2 and Yan's methods. All the results show that the proposed method is superior to the other two. In the following, three experiments with two different data sets are described. The first and second experiments were performed on an artificial three-class data, and the third on a large collection of handwritten numeral data.

Since all prototype optimization methods still contain undefined parameters, they should be defined first. For the results presented, A_1 and A_2 used in Yan's method are set to 1 and -1 respectively, and $\alpha(n)$ and $\eta(n)$ are

$$\alpha(n+1) = 0.995 * \alpha(n) \quad , \text{ with } \begin{cases} \alpha(0) = 0.01 & \text{for LVQ2;} \\ \alpha(0) = 0.01 & \text{for Yan's method;} \end{cases}$$

and

$$\eta(n+1) = 0.995 * \eta(n) \quad , \text{ with } \eta(0) = 0.1.$$

The given initial values of $\alpha(0)$ and $\eta(0)$ are derived from the following Experiment 1 so that the best classification results of the three prototype optimization methods are

obtained, respectively. However, it should be mentioned that the proposed method always obtains the best classification results during our experiments when $\alpha(0) = \eta(0)$. As for $A(n)$, it is expected to become small when n is large. However, since $A(n)$ stands for the receptive field where prototype update could take place, it should be correlated to the appropriateness of the corresponding prototypes, that is if the current prototypes are suitably located, $A(n)$ will be small, otherwise $A(n)$ will be large. To achieve both purposes, $A(n)$ is designed as

$$A(n) = \beta(n) * V'(n) \tag{72}$$

$$\beta(n) = 0.995 * \beta(n-1)$$

and

$$V'(n) = \frac{1}{Total\_no} \sum_{j=1}^{J} \sum_{i=0}^{I_j} \| x_i^j - r_j \|^2$$

where $\beta(0) = 1$, $Total\_no$ is the total number of training samples, J is the total number of prototypes, I_j corresponds to the prototype j of class k and stands for the number of those samples of class k, each of which has the minimum distance to prototype j among all prototypes of class k, and x_i^j is the feature vector of the ith sample belonging to prototype j. Heuristically, if the current prototypes are good, then $V'(n)$ tends to be small. The purpose to have $\beta(n)$ in $A(n)$ is to ensure that $A(\infty)$ becomes zero. For the sake of simplicity, different classes are assumed to possess the same number of prototypes in the following experiments.

Experiment 1: Prototype Initialization by k-means Algorithm

This experiment was performed on an artificial data set as shown in Figure 22, which contains three classes of 2-D planar points. There are 525 samples in total. The samples of the first, second, and third classes are represented by symbols '+', 'o', and '.', respectively. Using 2-D patterns enables a visual inspection of prototype movement and the corresponding classification boundary, so that the intrinsic characteristics of the three methods can be observed. The objective of this experiment is to compare the three prototype optimization methods in the context that initial prototypes are carefully selected, *i.e.* constructed by the k-means algorithm.

For a general comparison, four different quantities of prototypes per class are used. For a given number m, each approach will first find m prototypes for each class, and then compute the total number of correctly classified samples according to these m prototypes by the nearest neighbor classifier. Table 21 lists the performances for $m \in \{2, 3, 4, 5\}$, which shows clearly that the present method produces the best performance, LVQ2 the second and Yan's method the third. Figure 23 displays the positions of the final prototypes derived by the three methods and the classification boundaries constructed by these prototypes, when m equals 3.

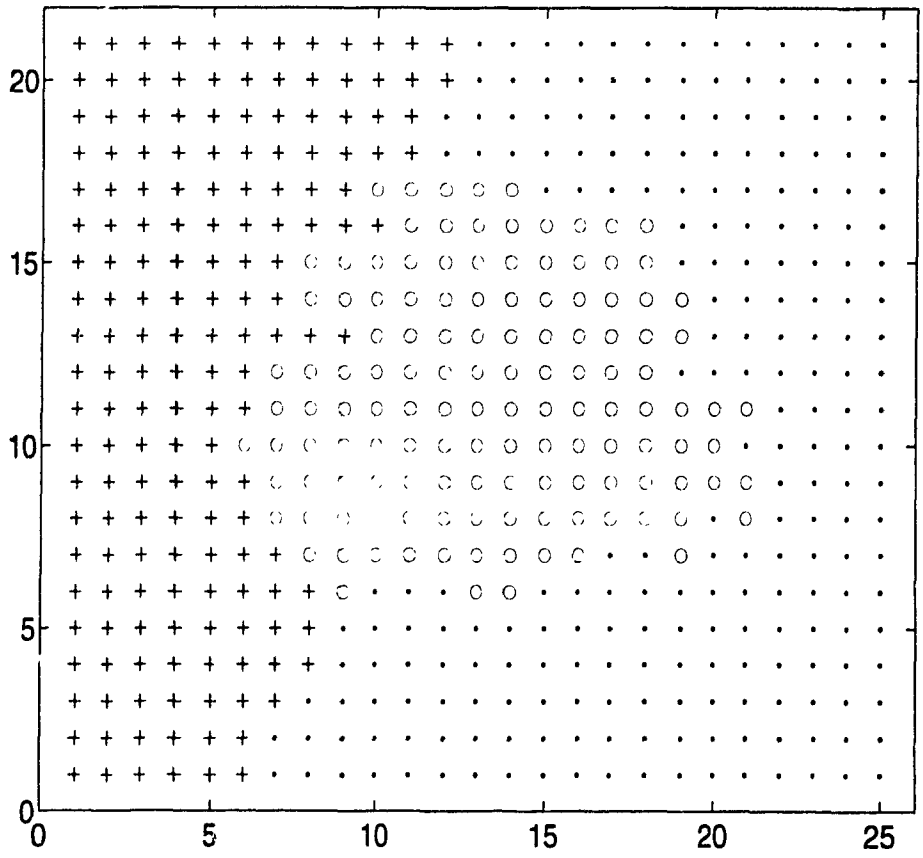


Figure 22: Distribution of samples of three classes.

Approach	m	Correct No.	Error No.
Yan	2	454	71
LVQ2	2	488	37
Present Method	2	495	30
Yan	3	475	50
LVQ2	3	494	31
Present Method	3	505	20
Yan	4	478	47
LVQ2	4	501	24
Present Method	4	511	14
Yan	5	480	45
LVQ2	5	508	17
Present Method	5	517	8

Table 21: Recognition performances by three prototype optimization methods, where m stands for the number of prototypes for each class, Correct No. for the number of correctly classified samples, and Error No. for the number of wrongly classified samples.

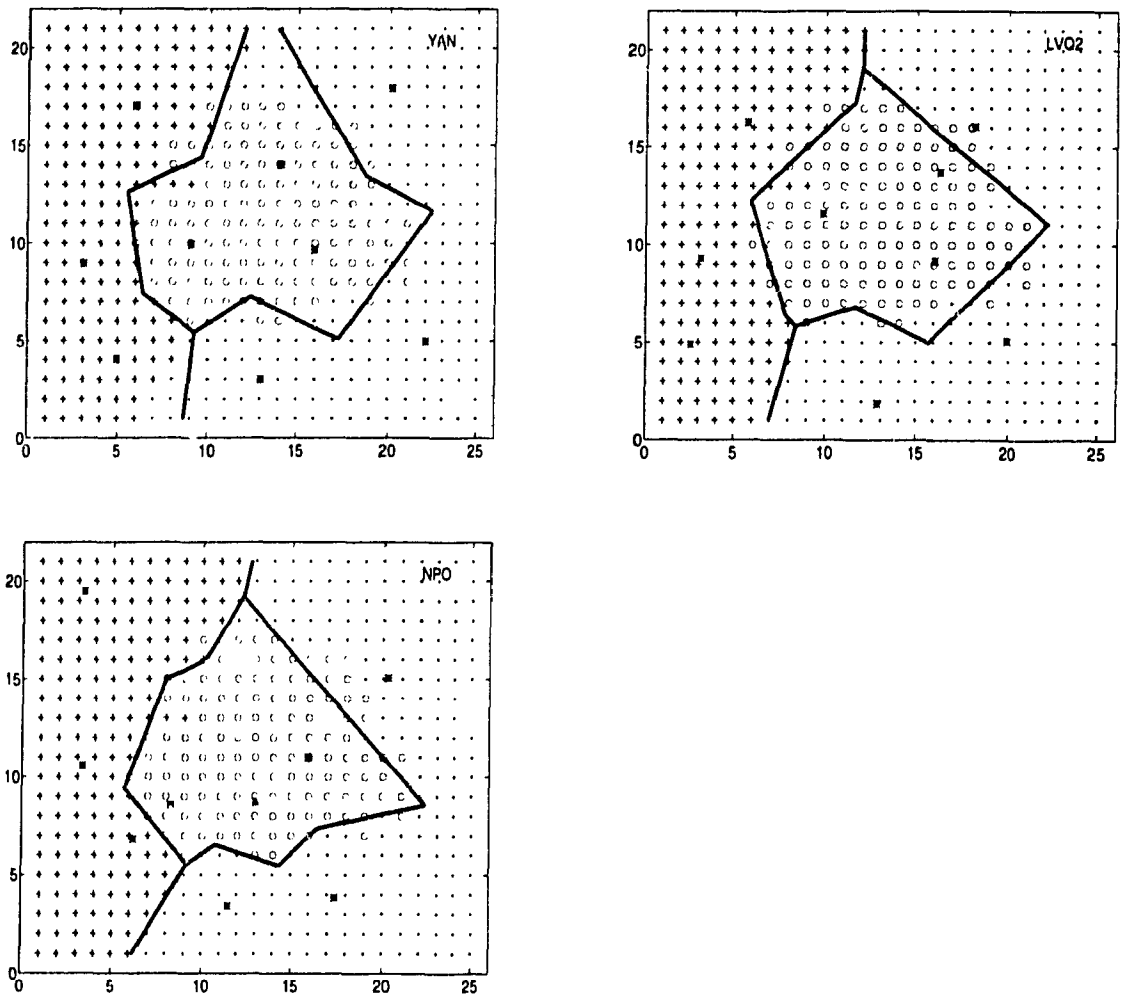


Figure 23: Prototypes and classification boundary constructed by the three prototype optimization methods. (NPO stands for the present method, and symbol * indicates the optimized prototype locations)

Experiment 2: Random Prototype Initialization

This experiment uses the same data as that used in Experiment 1, and the number of prototypes for each class is set to be 3. Instead of carefully selecting the initial prototypes, two random initializations are adopted here. The first is to select the first three encountered samples in each class set as the corresponding initial prototypes, and the second is to set all the three initial prototypes at the same point, the center of each class. The two initializations are shown in Figure 24. After training, Table 22 lists their classification results. Obviously, for both initializations, the present method performs much better than the other two, and it achieves almost the same classification performance as it has achieved in Experiment 1 for the same number of prototypes. This indicates that the present method is likely capable of constructing reasonable prototypes for arbitrary prototype initializations[3]. The reason why the present method can avoid being trapped at local minima is because of the intrinsic annealing property of its *stochastic-like* error function. This will become clear after showing two figures: Figures 25 and 27. Figure 25 displays the prototype movement in each iteration of the present method with the second prototype initialization. It shows that the optimized prototypes of each class diverges from the center of the corresponding class and gradually converges at their individual optimal locations. It is not easy to see that there are three converged locations for the first class because

[3]However, to confirm the hypothesis that the present method will not converge at local minima in general cases indeed needs many more experiments and the support from theoretic work.

Approach	Initialization 1		Initialization 2	
	Correct No.	Error No.	Correct No.	Error No.
Yan	361	164	389	136
LVQ2	417	108	476	49
Present Method	502	23	504	21

Table 22: Classification performances by the three prototype optimization approaches, where Correct No. stands for the number of correctly recognized samples, Error No. for the number of wrongly recognized samples, and each class contains three prototypes.

two of them are close to each other. The final prototypes and their corresponding classification boundary are shown in Figure 26, where the nine optimized prototypes can be seen clearly. Figure 27 shows the distribution of E_{new} and the classification error by the nearest neighbor classifier, where x-axis is the number of iterations, y-axis is the magnitude of error, the dotted line shows the distribution of E_{new}, and the solid line gives a distribution of the classification error by the nearest neighbor classifier. As illustrated in Section 3, E_{new} gradually approaches the classification error by the nearest neighbor classifier as n increases or $A(n)$ decreases, and finally they will converge at the same value. Interestingly, this distribution of E_{new} seems to show a stochastically decreasing error, starting at a large value and going down.

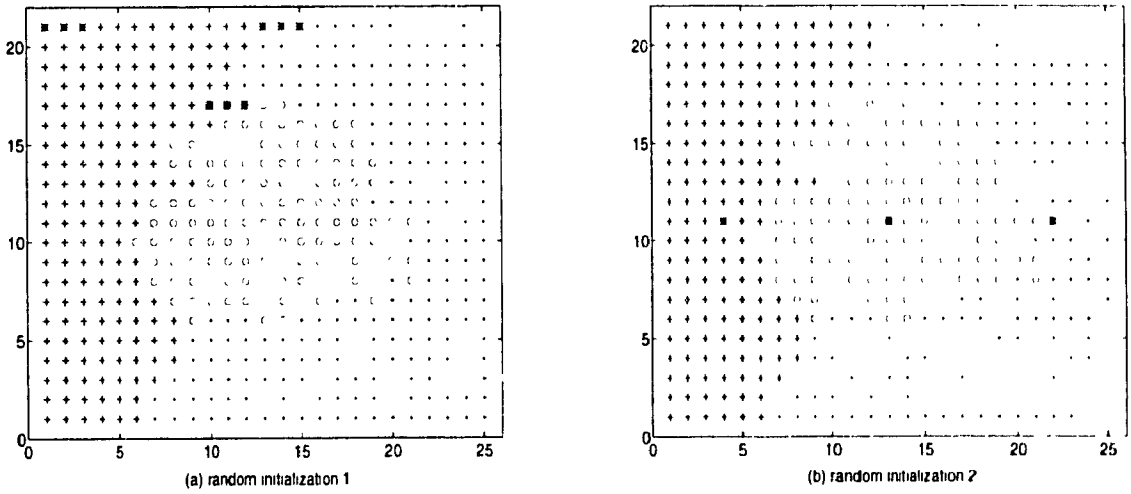


Figure 24: Display of two random initial prototypes, where symbol * indicates the locations of the initial prototypes. (a) random initialization 1: the first three encountered samples in each class set are selected to be the corresponding initial prototypes, and (b) random initialization 2: all the three initial prototypes are set at the same point, the center of each class.

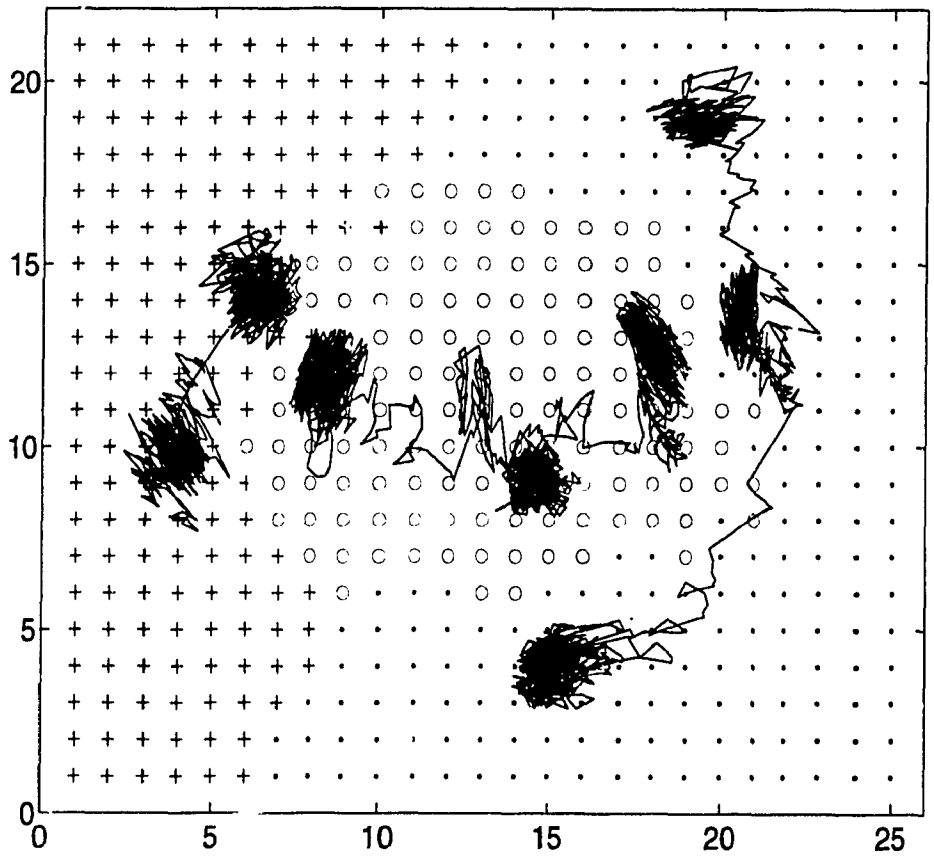


Figure 25: The trace of prototype movement derived by the present method with initialization of the second random prototypes.

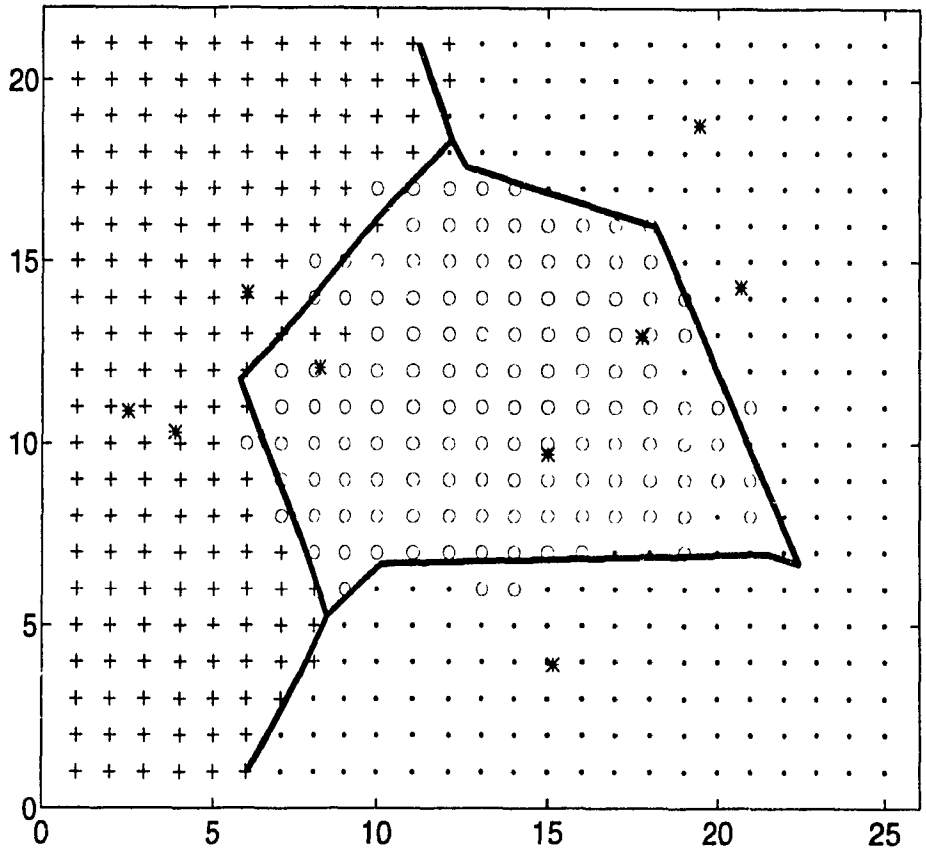


Figure 26: Optimized prototype and boundary positions derived by the present method with initialization of the second random prototypes.

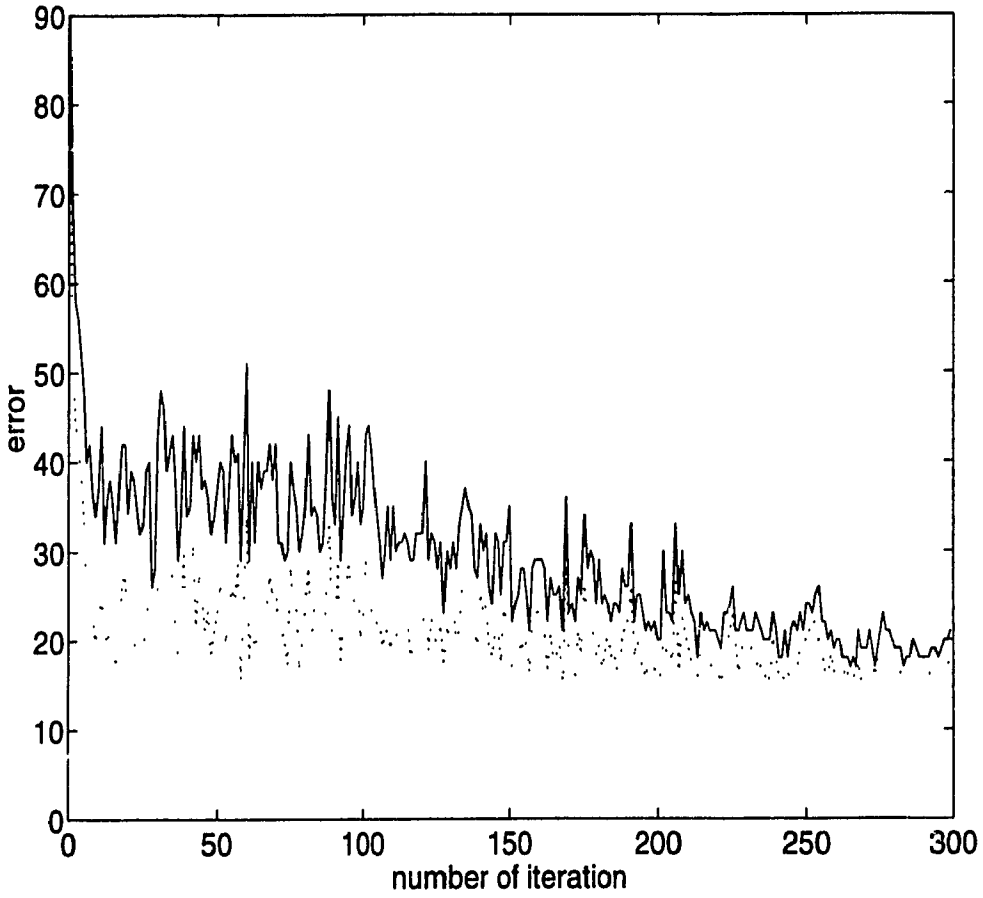


Figure 27: Distribution of E_{new} and the classification error by the nearest neighbor classifier, where x-axis is the number of iterations, y-axis is the magnitude of error. The distribution of E_{new} is indicated by the dotted line, and the solid line shows the distribution of the classification error by the nearest neighbor classifier.

Experiment 3: Prototype Optimization on Handwritten Numeral Data

The purpose of this experiment is to investigate the performance of the prototype optimization method on real data, and the generalization of the optimized prototypes to unseen patterns. The data performed in this experiment consist of the ITRI's numeral data base. Sets 1-4 (19,353 samples) were used for prototypes training, and Sets 5-9 (22,024 samples) for performance testing. Table 23 lists the recognition accuracy of the three classifiers on the testing data set (22,024 samples). For each individual classifier, it classifies each sample and produces K, *i.e.* the total number of classes, measurement values, each of which stands for the degree that this sample belongs to one particular class; then the measurement values produced by all individual classifiers on each sample are constituted into a feature vector. As a result, each sample contains one feature vector, which represents the location of this sample in the corresponding feature space. With the three prototype optimization methods and five prototypes for each class, Table 24 shows the classification accuracy on the training and testing data sets respectively. For both the training and testing data, the present method again obtains the best recognition result, however LVQ2 shows a compatible performance in this experiment.

	Rec.	Sub.	Rel.
e_1	89.77	10.23	0.8977
e_2	90.50	9.50	0.9050
e_3	91.68	8.32	0.9168

Table 23: Recognition performances of individual classifiers using 22,024 testing samples.

	Learning		Testing	
	Rec.	Sub.	Rec.	Sub.
Yan	90.34	9.66	88.70	11.30
LVQ2	98.52	1.48	96.61	3.39
present method	98.83	1.17	96.91	3.09

Table 24: Classification performances of three prototype optimization methods on 22,024 testing numeral patterns by using the recognition output of the three individual classifiers.

Chapter 9

Conclusion

9.1 Summary

The following statements appeared in Kanal[1] [134]:

> *"It is now recognized that the key to the pattern recognition problem does not lie wholly in learning machines, statistical approaches, formal linguistic approaches, or in any other particular solution which has been vigorously advocated by one or another group during the last one and half decades as the solution to the pattern recognition problem. No single model exists for all pattern recognition problems and no single technique is applicable to all problems. Rather what we have in pattern recognition is a bag of tools and a bag of problems."*

[1] Prof. Kanal is the recipient of the 1992 King-Sun Fu Award.

Later in the same paper he mentioned:

> *"What is incumbent on us is to attempt to understand the capabilities and applicability of the various tools and exploit the complementary advantages of the different paradigms."*

Indeed, Kanal tried to argue that the research addressed to the problem of combining multiple classifiers may provide new insight into pattern recognition. Previously, the main efforts focused on the design of one good classifier, so that a desired classification rate could be obtained. Now there is a different focus. Instead of designing one high-performance classifier (which is extremely difficult), we can build a number of different and complementary ones. Each classifier itself may not have a superb performance; however, the appropriate combination of these individual classifiers may produce a highly reliable performance [51].

In fact, the idea of CME is not new: in human society, when people encounter a complicated problem, they tend to assign a group of experts to solve it. This is because experience has shown that group decision in general is better than any individual's. However, there are three basic requirements for constituting a successful group: (1) there exist several individuals or experts who are qualified to deal with the problem; (2) the chosen experts are both able and willing to cooperate with each other; and (3) there exists an efficient and effective decision-making mechanism, which can resolve the potential conflicts in the group and achieve the final consensus. Interestingly,

majority voting seems to be the most commonly used method to derive a consensus in human communities. Although voting is fair in the sense that each individual has an equal right to make his or her decision, it will not be the best approach for making the right decision if some individuals have shown themselves capable of making more correct decisions than others.

For character recognition, to date, a large number of character features and classification functions have been developed. Therefore, the first requirement of successful teamwork, the availability of expert-like classifiers, is satisfied. In general, the output information that various classifiers supply can be divided into two levels: (1) The abstract level: a classifier only outputs a unique class label, or a subset of class labels when it cannot decide on the identities of some confusing patterns; and (2) The measurement level: a classifier assigns each class label a measurement value, to indicate the degree of possibility that the corresponding class pertains to the input pattern. Classifiers producing information at the abstract or measurement levels are called *abstract-level* or *measurement-level classifiers*. The combination of abstract-level or measurement-level classifiers is called *abstract-level* or *measurement-level CME*. Because abstract- and measurement-level CME make use of different levels of information, they naturally require different combination functions to derive the best classification decision.

For abstract-level CME, a novel combination model is proposed in this research, the Behavior-Knowledge Space method (the BKS method). This method operates in two stages: (1) the knowledge-modelling stage, which extracts knowledge from the former classifiers' decisions and constructs a K-dimensional behavior-knowledge space; and (2) the decision-making stage, which is carried out for each test sample, and which combines decisions generated from individual classifiers, and selects a specific cell of the constructed space to make a final decision by a rule utilizing the knowledge of the cell. It has been shown that the BKS method possesses many advantageous properties, such as (1) adaptive learning, (2) automatic threshold finding, (3) theoretical performance analysis of the combination of partial classifiers, (4) the CME method to achieve the highest recognition accuracy for a given set of abstract-level classifier from the statistical point of view, (5) semi-monotonicity improvement in recognition accuracy with respect to the increment of number of combined classifiers, and (6) no assumption that classifiers are independent of each other and no degradation when dependence exists among classifiers. However, the BKS method indeed has two intrinsic problems which may considerably constrain its effectiveness: not enough learning samples, and exponential memory requirements. Fortunately, with the help of another abstract-level combination function, the first problem can be effectively resolved. Also, three solutions have been proposed for the second problem: (1) dynamic class allocation, (2) dynamic cell allocation, and (3) condensed

editing technique. With three classifiers, experiments on ITRI's numeral database have shown that the BKS method outperforms the other three abstract-level combination models: voting, Bayesian, and Dempster-Shafer. The good performance of the BKS method has also been reported in different resources such as [135, 136]. This reveals that this method is practical to many OCR applications.

For measurement-level CME, since classifiers may produce measurement values with different physical meanings and scales, before combining the measurement values they should be transformed into a new data form with the same meaning (such as the larger the better) and the same scale (such as [0.0 - 1.0]). As a matter of fact, the data transformation process is to ensure that the second requirement of a successful group can be satisfied. Three combination approaches have been proposed in this thesis: LCA, BCA and neural network models. The first two approaches (LCA and BCA) transform a measurement value into a reliability-like expression in terms of conditional probability. Then the transformed measurement values are either summed up in a linear way, or multiplied together through the Bayesian formula. Implicitly, both approaches require a few assumptions of the characteristics or the distributions among measurement values or classifiers which significantly reduce their effectiveness in real applications. Looking at the CME problem from a different point of view, for a multiple-classifier system, individual classifiers can be regarded as feature extractors as well. With this new understanding, neural network approaches become applicable

to serve the CME combination functions. Intrinsically, neural networks are suitable for measurement-level CME because they posses four well-known valuable characteristics: (1) they behave as *collective systems*; (2) they can infer subtle, unknown relationships from data; (3) they can generalize, meaning that they can respond correctly to patterns that are similar to the original training data; and (4) they are nonlinear, that is, they can solve some complex problems more accurately than linear techniques do. Amazingly, these characteristics have overcome all the constraints presented in either LCA or BCA. As a matter of fact, the four characteristics specify exactly the desired functions of CME. Since CME is a supervised learning problem, the multi-layer perceptron with back-propagation error correction is adopted to combine all measurement values, because it has been used successfully in various pattern recognition applications with good recognition results. In this thesis, a family of data transformation functions are proposed which are capable of being applied to most kinds of measurement values. To further improve both speed and generalization of a multi-layer perceptron, three strategies have been proposed: training by boundary samples, training by partitions, and weight reduction. With three classifiers, experiments on ITRI's numeral database show that by using a three-layer perceptron, the recognition rate can be improved up to 97.66% with no rejection (for the same testing samples, the best recognition rate of the three classifiers is 91.68%).

Essentially, CME deals with classifiers that use different types of features and different classification techniques; however, CME can become a new classifier design technique as well. In this thesis, two classification procedures based on CME are proposed: recognition by parts and recognition by pair classifiers. Both of them first use the same types of features and classification functions to construct several classifiers, and then use CME techniques to combine the measurement-level output information of all constructed classifiers together. Simply speaking, recognition by parts divides high-dimensional feature vectors into several lower-dimension feature vectors, each of which represents the features of a sub-part of image and is used to construct a so-called sub-part classifiers. This is because high-dimensional feature vectors do not only increase computational complexity but also produce implementation and accuracy problems. Recognition by pair classifiers implements one classifier for each pair of classes. Therefore, for an M-class recognition problem there are $\frac{M*(M+1)}{2}$ pair classifiers, each of which has a high discrimination capability for recognizing a pair of classes. Naturally, CME techniques are required to integrate all classification decisions of sub-part or pair classifiers. Experiments have shown promising recognition performance by using these two classification procedures.

Since the basic component of a multiple-classifier system is a classifier and CME turns out to be a generic pattern recognition problem, it is essential to improve the recognition efficiency of classifiers. In this thesis, we also made the effort to derive

the optimized prototypes for the nearest neighbor classifier. It is well known that the nearest neighbor classifier is one of the most commonly-used classifiers due to its simplicity and discriminant capability. However, the nearest neighbor classifier intrinsically possesses two built-in disadvantages: (1) Outlying samples[2] will affect its recognition accuracy considerably, and (2) It is computationally expensive when the number of training samples is fairly large. A way to solve the problems is by prototype optimization. Using a few but well optimized prototypes of which the number is considerably smaller than the total number of training samples, it is possible to achieve an even better performance in both speed and accuracy than using all the training samples. In Chapter 8, a novel approach for prototype optimization is proposed based on a neural network technique. A new network architecture is designed for optimizing prototypes and a new error function is proposed for training the network. Promising results indicate that the proposed error function can truly correspond to the classification error by a nearest neighbor classifier. There are two main characteristics of the present method: (1) consistent criteria, for updating the weights of the network which correspond to the prototypes to be optimized, and for using the trained prototypes to build a nearest neighbor classifier; (2) the derived prototype update rule possesses a deterministic annealing property, hence the minimization the proposed error function can likely avoid convergence at local minima.

[2]Outlying samples of one class are the samples which are far away from most samples of this class. Usually, they are rare and irregular.

One important conclusion is that LVQ2 can be considered as a special case of the proposed method. The experimental results for the comparison of new method, Yan's method and LVQ2 show that the present method is superior to the other two.

9.2 Future Directions

The combination of multiple classifiers, in fact, is not only useful to the recognition of handwritten numerals, but also to various application areas of pattern recognition (*e.g.*, fingerprint recognition, face recognition, and medical diagnosis). Although several combination models have been proposed, many new problems have been brought out for further study as well, and listed below as challenging open problems:

1. For abstract-level classifiers, from the statistical point of view the BKS method can produce the best recognition accuracy. For measurement-level classifiers, neural networks have been shown capable to combine their classification results together effectively. But, for an application with both abstract-level and measurement-level classifiers, it is natural to ask: what are the effective combination functions? If enough representative training samples are available, one possible solution is first to combine all measurement-level classifiers with a neural network, and regard the neural network as an abstract-level classifier by only outputting the class label of its first choice with an information reduction

process; then all abstract-level classifiers including the neural network are combined by using the BKS method. The main reason to adopt this combination scheme is that the BKS method contains the semi-monotonicity property, which ascertains that the recognition performance of the BKS method is equal to or better than any abstract-level classifier, including the neural network. However, more research effort and experiments need to be performed on this issue.

2. All advantageous properties of the BKS method exist from the statistical point of view. This means that a large enough and representative learning data set should be provided. If only a few samples are collected, or samples are collected randomly and carelessly, the desired properties of this method cannot be guaranteed. Therefore, for practical applications, the key issue to successfully apply this method is to construct a representative training data base. This indicates that more attention should be paid to the data collection step, which is often ignored in the current research domain. Fenrich and Hull [137] have presented the concerns in the creation of an image database.

3. Although measurement-level classifiers usually supply positive measurement values, it is not ruled out that some classifiers may supply negative measurement values (*e.g.* distance through logarithmic functions). This brings out one question: when there are both positive and negative measurement values, how and what are the appropriate data transformation functions which can transform

them into the same likeness form?

4. Training by partition has shown its effectiveness through our experiments. However, are there other criteria which can better, or even best, partition the pattern space S? If so, what are they? In general, such criteria may include features in the input patterns as well as characteristics of classifier outputs.

5. In Chapter 1, we mentioned that there are two key tasks in CME research, and till now we have only focused on the second task, the combination models. In fact, the first task is also important for achieving high recognition and reliability systems. Accordingly, much effort should be spent on how to systematically construct or select the features and classification methodologies which can compensate for one another and get the best combination result. Some research on this topic has been directed by Kleinberg [138, 139].

6. Since the proposed error function in Equation (64) is closely related to the real error of classification, it in fact can be applied to various classification problems based on the minimization of an error function. Currently we are applying the new error function on multi-layer perceptrons and radial-basis function neural networks.

Due to the promising results obtained from various experiments, we believe that CME is one of the key technologies for developing practical systems of pattern recognition which are capable of matching human performance. We are eager to see that more and more research effort is devoted to this fascinating topic.

References

[1] D. Noton. "A Theory of Visual Pattern Perception," *IEEE Transactions on Systems Science and Cybernetics,* SSC-6, No. 4, pp. 349–357, 1970.

[2] D. Noton and L. Stark. "Eye Movements and Visual Perception," *Scientific American,* Vol. 224, No. 6, pp. 34–43, 1971.

[3] M. Nadler and E.P. Smith. Pattern Recognition Engineering, John Wiley & Sons, New York, 1993.

[4] C.Y. Suen. "Distinctive Features in Automatic Recognition of Handprinted Characters," *Signal Processing,* Vol. 4, pp. 193–207, 1982.

[5] M. Shimura. "Multicategory Learning Classifiers for Character Reading," *IEEE Trans. Syst., Man, Cybern.,* Vol. 3, pp. 74–85, 1973.

[6] W.W. Bledsoe and I. Browning. "Pattern Recognition and Reading by Machine," *Proc. EJCC,* pp. 225-232. 1959.

[7] S.K. Kwon and D.C. Lai. "Recognition Experiments with Handprinted Numerals," *Proc. Workshop on Pattern Recognition and Artificial Intelligence,* pp. 74–83, 1976.

[8] B.A. Glucksman. "Classification of Mixed-Font Alphabetics by Characteristic Loci," *Dig. 1st Ann. IEEE Comput. Conf.,* pp. 138-141, Sept., 1991.

[9] R.C. Gonzalez and P. Wintz. Digital Image Processing, 2nd ed., Addison-Wesley, Reading, Ma, 1987.

[10] W.K. Pratt. Digital Image Processing, 2nd ed., Wiley, New York, 1991.

[11] H.C. Andrews. "Multi-Dimensional Rotations in Feature Selection," *IEEE Trans. Comput.,* Vol. 20, pp.1045–1051,1971.

[12] J.S. Huang and M.L. Chung. "Separating Similar Complex Chinese Characters by Walsh Transform," *Pattern recognition,* Vol. 4, No. 4, pp.425–428, 1987.

[13] P.A. Devijver and J. Kittler. Pattern Recognition — A Statistical Approach, London: Prentice-Hall, 1982.

[14] R.O. Duda. "Elements of Pattern Recognition," in: J.M. Mendal and K.S. Fu, eds., *Adaptive Learning and pattern Recognition Systems,* Academic Press, New York, 1970.

[15] C.Y. Suen and R.J. Shillman. "Low Error Rate Optical Character Recognition of Unconstrained Handprinted Letters Based on a Model of Human Perception," *IEEE Trans. Syst., Man, Cybern.*, Vol. 7, pp. 491–495, 1977.

[16] L.G. Roberts. "Machine Perception of Three-Dimensional Solids," in *Optical and Electro-Optical Information Processing*, (J.T. Tippet, ed.), MIT Press, Cambridge, Mass, 1965.

[17] M. Nadler. "A Note on the Coefficients of Compass Mask Coefficients," *Computer Vision, Graphics, and Image Processing*, Vol. 51, pp. 96–101, 1990.

[18] L. Lam, S.W. Lee, and C.Y. Suen. "Thinning Methodologies - A Comprehensive Survey," *IEEE Trans. Patt. Anal. Machine Intell.*, Vol. 14, No. 9, pp. 869-885, 1992.

[19] T.M. Cover and P.E. Hart. "Nearest Neighbor Pattern Classification," *IEEE Trans. Inform. Theory*, Vol. 13, pp. 21–27, 1967.

[20] P.E. Hart. "The Condensed Nearest Neighbor Rule," *IEEE Trans. Inform. Theory*, Vol. 14, pp. 515-516, 1968.

[21] M.L. Mico, J. Oncina, and E. Vidal. "A New Version Of The Nearest-Neighbour Approximating And Eliminating Search Algorithm (AESA) With Linear Preprocessing Time And Memory Requirements," *Pattern Recognition Letters*, Vol. 4 No. 3, pp. 9–18, 1994.

[22] C.L. Chang. "Finding Prototypes For Nearest Neighbor Classifiers," *IEEE Trans. Comput.*, Vol. 23, pp. 1179–1184, 1974.

[23] H. Yan. "Prototype Optimization For Nearest Neighbor Classifiers Using A Two-layer Perceptron," *Pattern Recognition*, Vol. 26, No. 2, pp. 317–324, 1993.

[24] D.S. Lee, S.W. Lam, and S.N. Srihari. "A Structural Approach to Recognize Hand-Printed and Degraded Machine-Printed Characters," *Pre-Proceedings of the IAPR Syntactical and Structural Pattern Recognition Workshop*, pp. 256–272, Murray Hill, NJ, June, 1990.

[25] K. Fukunaga. Introduction to Statistical Pattern Recognition, Academic Press, New York, 1972.

[26] R.O. Duda and P.E. Hart. Pattern Classification and Scene Analysis, Addison-Wesley, New York, 1973.

[27] F. Rosenblatt. Principles of Neurodynamics: Perceptrons and The Theory of Brain Mechanisms, Spartan, New York, 1962.

[28] J. Schürmann and W. Doster. "A Decision Theoretic Approach to Hierarchical Classifier Design," *Pattern Recognition*, Vol. 17, No. 3, pp. 359–369, 1984.

[29] N. Chomsky. "Three Models for the Description of Language," *Transactions on Information Theory*, IT-2, No. 3, pp. 113–124, 1956.

[30] N. Chomsky. "On Certain Formal Properties of Grammars," *Information and Control*, Vol. 2, No. 2, pp. 137–167, 1959.

[31] N. Chomsky. "Formal Properties of Grammars," in: *Handbook of Mathematical Psychology*, Vol. 2, New York, Wiley, pp. 323–418, 1963.

[32] K.S. Fu and T.L. Booth. "Grammatical Inference: Introduction and Survey," *IEEE Trans. Systems Man Cybernet.*, Pt I, SMC-5, No. 1, pp. 95–111, 1975; Pt II, No. 4, pp. 409–423, 1975.

[33] K.S. Fu. Syntactic Pattern Recognition and Applications, Prentice-Hall, Englewood Cliffs, NJ, 1982.

[34] H. Yamada. "Contour DP Matching Method and Its Applications to Hand-Printed Chinese Character Recognition," *Proc. 7 th IJCPR*, pp. 389-392, 1984.

[35] R.P. Lippmann. "An Introduction to Computing with Neural Nets," *IEEE ASSP Magazine*, pp. 4–22, April 1987.

[36] B. Widrow and R. Winter. "Neural Nets for Adaptive Filtering and Adaptive Pattern Recognition," *Computer*, Vol. 21, No. 3, pp. 25–39, March 1988.

[37] Bart Kosko. "Bidirectional Associative Memories," *IEEE Trans. Syst., Man, Cybern.*, Vol. SMC-18, No. 1, pp. 49–60, 1988.

[38] J.J. Hopfield and D.W. Tank. "Computing with Neural Circuits: A Model," *Science*, Vol. 233, pp. 625-633, August 1987.

[39] D.H. Ackley, G.E. Hinton, and T.J. Sejnowski. "A Learning Algorithm for Boltzmann Machines," *Cognitive Science*, Vol. 9, pp. 147-169, 1985.

[40] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. "Optimization by Simulation Annealing," *Science*, Vol. 220, pp. 671-680, 1983.

[41] R. Hecht-Nielsen. "Counterpropagation Networks," *Applied Optics,* Vol. 26, No. 23, pp. 4979-4984, 1987.

[42] T. Kohonen. Self-Organization and Associative Memory, volume 8 of Springer Series in Information Sciences, Springer-Verlag, New York, 1988.

[43] G.A. Carpenter and S. Grossberg. "The Art of Adaptive Pattern Recognition by a Self-Organizing Neural Network," *Computer,* Vol. 21, No. 3, pp. 77-88, March 1988.

[44] K. Fukushima, S. Miyake, and T. Ito. "Neocognitron: A Neural Network Model for a Mechanism of Visual Pattern Recognition," *IEEE Trans. Syst., Man, Cybern.,* Vol. SMC-13, No. 5, pp. 826-834, 1983.

[45] T. Poggio and F. Girosi. "Regularization Algorithms for Learning That Are Equivalent to Multilayer Networks," *Science*, Vol. 247, pp. 978-982, 1990.

[46] C.Y. Suen, C. Nadal, R. Legault, T.A. Mai, and L. Lam. "Computer Recognition of Unconstrained Handwritten Numerals," *Proc. IEEE*, Vol. 80, No. 7, pp. 1162 1180, 1992.

[47] B. Duerr, W. Haettich, H. Tropf, and G. Winkler. "A Combination of Statistical and Syntactical Pattern Recognition Applied to Classification of Unconstrained Handwritten Numerals," *Pattern Recognition*, Vol. 12, pp. 189-199, 1980.

[48] E. Mandler and J. Schürmann. "Combining The Classification Results of Independent Classifiers Based on the Dempster-Shafer Theory of Evidence," in *Pattern Recognition and Artificial Intelligence*, Gelsema and Kanal, Eds. Amsterdam: Elsevier Science, North-Holland, pp. 381-393, 1988.

[49] X. Ling and W.G. Rudd. "Combining Opinions from Several Experts," *Applied Artificial Intelligence*, Vol. 3, pp. 439-452, Hemisphere Publishing Corporation, 1989.

[50] J.J. Hull, A. Commike, and T.K. Ho. "Multiple Algorithms for Handwritten Character Recognition," *Proc. International Workshop on Frontiers in Handwritten Recognition*, pp. 117-129, 1990, Montréal, Canada.

[51] C. Nadal, R. Legault, and C.Y. Suen. "Complementary Algorithms for the Recognition of Totally Unconstrained Handwritten Numerals," *Proc. International Conference on Pattern Recognition*, Vol. 1, pp. 443-449, 1990, Atlantic City, New Jersey, USA.

[52] C.Y. Suen, C. Nadal, T.A. Mai, R. Legault, and L. Lam. "Recognition of Totally Unconstrained Handwritten Numerals Based on the Concept of Multiple Experts," *Proc. International Workshop on Frontiers in Handwritten Recognition,* pp. 131-143, 1990, Montréal, Canada.

[53] J.D. Tubbs and W.O. Alltop. "Measures of Confidence Associated with Combining Classification Results," *IEEE Trans. Systems Man Cybernet.,* Vol. 21, No. 3, pp. 690–692, 1991.

[54] L. Xu, A. Krzyzak, and C.Y. Suen. "Methods of Combining Multiple Classifiers and Their Application to Handwritten Numeral Recognition," *IEEE Trans. on Systems, Man and Cybernetics,* Vol. SMC-22, No. 3, pp. 418–435, 1992.

[55] C.Y. Suen, R. Legault, C. Nadal, M. Cheriet, and L. Lam. "Building a New Generation of Handwriting Recognition Systems," *Pattern Recognition Letters,* Vol. 14, No. 4, pp. 303–315, 1993.

[56] D.S. Lee and S.N. Srihari. "Handprinted Digit Recognition: A Comparison of Algorithms," *Pre-Proc. International Workshop on Frontiers in Handwriting Recognition,* pp. 153–162, Buffalo, New York, USA, 1993.

[57] F.F. Soulie, E. Vinnet, and B. Lamy. "Multi-Modular Neural Network Architectures: Applications in Optical Character and Human Face Recognition," *Int. Journal of Pattern Recognition and Artificial Intelligence,* Vol. 5, No. 4, pp. 721–755, 1993.

[58] H.P. Nii. "Blackboard Systems," AI Magazine, 7(2,3): 38-53, 82-106, 1986.

[59] R. Dodhiawala, V. Jagannathan, and L.S. Baum. Blackboard Architectures and Applications, Academic Press, Boston, 1989.

[60] L. Lam and C.Y. Suen. "Structural Classification and Relaxation Matching of Totally Unconstrained Handwritten Zip-Code Numerals," *Pattern Recognition*, Vol. 21, No. 1, pp. 19-31, 1988.

[61] C.L. Kuan and S.N. Srihari. "A Stroke-Based Approach to Handwritten Numeral Recognition," *Proc. US Postal Service Adv. Techn. Conf.*, pp. 1033-1041, 1988.

[62] F. Kimura and M. Shridhar. "Handwritten Numerical Recognition Based on Multiple Algorithms," *Pattern Recognition*, Vol. 24, No. 10, pp. 969-983, 1991.

[63] R.E. Bellman. Dynamic Programming, Princeton University Press, Princeton, NJ, 1957.

[64] D. Black. The Theory of Committees and Elections, Cambridge University Press, London, 1958.

[65] K.J. Arrow. Social Choice and Individual Values, Wiley, New York, 1963.

[66] P.C. Fishburn. The Theory of Social Choice, Princeton University Press, Princeton, 1973.

[67] H. Einhorn. "Expert Judgment: Some Necessary Conditions and an Example," *Journal of Applied Psychology*, Vol. 59, No. 5, pp. 562–571, 1974.

[68] D.E. O'Leary and K.V. Pincus. Models of Consensus for Validation of Expert System, School of Business, University of Southern California, Los Angeles, CA 90089-1421, February, 1992.

[69] A.A. Mongi and C.G. Ralph. Data fusion in Robotics and Machine Intelligence, Academic Press, Boston, 1992.

[70] M.D. McLeish and M. Cecile. "Enhancing Medical Expert Systems with Knowledge Obtained from Statistical Data," *Annals of Mathematics and Artificial Intelligence,* Vol. 2, pp. 261–276, 1990.

[71] M.D. McLeish, P. Yao, and T. Stirtzinger. "A Study on the Use of Belief Functions for Medical Expert Systems," *Journal of Applied Statistics,* Vol. 18, No. 1, pp. 155–174, 1991.

[72] N.D. Clarke, M.D. McLeish, and T.J. Vyn. "Using Certainty Factors and Possibility Theory Methods in a Tillage Selection Expert System," *Expert Systems with Applications*, Vol. 4, pp. 53-62, 1992.

[73] J.C. Borda. "Mémoire sur les élections au scrutin," Hist. Acad. Royale Sci, 1781.

[74] F.S. Roberts. Discrete Mathematical Models with Applications to Social, Biological, and Environmental Problems, Prentice-Hall, Englewood Cliffs, NJ, 1976.

[75] G. Shafer and R. Logan. "Implementing Dempster's Rule for Hierarchical Evidence," *Artificial Intelligence*, Vol. 33, pp. 271-298, 1987.

[76] G. Shafer. A Mathematical Theory of Evidence, Princeton University Press, Princeton, 1976.

[77] B.G. Buchanan and E.H. Shortliffe. Rule-Based Expert Systems The MYCIN Experiments of the Stanford Heuristic Programming Project, Addison-Wesley, Reading, MA, 1984.

[78] J. Pearl. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann Publishers, Inc., San Mateo, California, 1988.

[79] R.O. Duda, P.E. Hart, N.J. Nilsson, R. Reboh, J.J. Slocum, and G. Sutherland. Development of a Computer-Based Consultant for Mineral Exploration, *SRI Annual Report,* SRI Projects 5821 and 6415, SRI International, Menlo Park, CA, 1977.

[80] L.A. Zadeh. "The Role of Fuzzy Logic in the Management of Uncertainty in Expert Systems," *Fuzzy Set and Systems,* Vol. 11, pp. 199 227, 1983.

[81] T. Kohonen. "The Self-Organizing Map," *Proc. of the IEEE,* Vol. 78, pp. 1468 1480, 1990.

[82] H. Yan. "Handwritten Digit Recognition Using as Optimized Nearest Neighbor Classifier," *Pattern Recognition Letters*, Vol. 15, No. 2, pp. 207–211, 1994.

[83] L. Xu, A. Krzyzak, and C.Y. Suen. "Associative Switch for Combining Multiple Classifiers," *Journal of Artificial Neural Networks*, Vol. 1, No. 1, pp. 77–100, 1994.

[84] S.J. Nowlan and G.E. Hinton. "Evaluation of Adaptive Mixtures of Competing Experts," In D.S. Touretzky, R. Lippman, (eds.) *Advances in Neural Information Processing System 3.* Morgan Kaufmann, San Mateo, CA., 1991.

[85] R.A. Jacobs, M.J. Jordan, S.J. Nowlan, and G.E. Hinton. "Adaptive Mixtures of Local Experts," *Neural Computation*, pp. 79– 87, Vol. 3, 1991.

[86] T.K. Ho. A Theory of Multiple Classifier Systems and Its Application to Visual Word Recognition, *Doctoral Dissertation, Department of Computer Science, State University of New York at Buffalo*, 1992.

[87] J. Franke and E. Mandler. "A Comparison of Two Approaches for Combining the Votes of Cooperating Classifiers," *Proc. 11th International Conference on Pattern Recognition*, Volume 2, pp. 611-614, 1992.

[88] J Franke. "On the Functional Classifier," *Proc. 1st Int. Conf. on Document Analysis and Recognition*, St. Malo, pp. 481–489, 1991.

[89] J. Franke. "Statistical Combination of Multiple Classifiers Adapted on Image Parts," *1st European Conference dedicated to Postal Technologies, JET POSTE 93, Nantes*, pp. 566–572, 1993.

[90] Y.S. Huang and C.Y. Suen. "The Recognition of Unconstrained Handwritten Numerals on a Multi-Classifier Space," submitted to *IEEE Trans. on Systems, Man and Cybernetics*, 1994.

[91] Y.S. Huang and C.Y. Suen. "The Behavior-Knowledge Space Method for the Combination of Multiple Classifiers," *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 347–352, New York, 1993.

[92] C.Y. Suen and Y.S. Huang. "Multi-Expert Systems for Pattern Recognition," *Proc. 2nd Pacific Rim International Conference on Artificial Intelligence*, pp. 15–20, 1992, Seoul, Korea.

[93] Y.S. Huang and C.Y. Suen. 'Recognition of Handwritten Numerals by Combining Multiple Experts," *Proc. Sixth International Conference on Handwriting and Drawing*, pp. 86–88, Paris, 1993.

[94] Y.S. Huang and C.Y. Suen. "An Optimal Method of Combining Multiple Experts for Handwritten Numerical Recognition," *Pre-Proc. International Workshop on Frontiers in Handwriting Recognition*, pp. 11–20, Buffalo, New York, USA, 1993.

[95] Y.S. Huang and C.Y. Suen. "A Knowledge Model with Decision Making for Combination of Multiple Classifiers," *the Fourth International Conference on Cognitive and Computer Science for Organization,* pp. 194–202, Montreal, Canada, 1993.

[96] Y.S. Huang and C.Y. Suen. "A Method of Combining Multiple Experts for the Recognition of Unconstrained Handwritten Numerals," *IEEE Trans. on Pattern Recognition and Artificial Intelligence,* accepted for publication.

[97] R.L. Klein and R.C. Dubes. "Experiments in Projection and Clustering by Simulated Annealing," *Pattern Recognition,* Vol. 22, No. 2, pp. 213–220, 1989.

[98] N. Baba. "A New Approach for Finding the Global Minimum Error Function of Neural Networks," *Neural Networks,* Vol. 2, pp. 367–373, 1989.

[99] P.L. Karlton, S.H. Fuller, R.E. Scroggs, and E.B. Koehler. "Performance of Height-Balanced Tree," *CACM,* Vol. 19, No. 1, pp. 23–28, 1976.

[100] B. Salzberg. File Structures – an Analytical Approach, Prentice Hall, Englewood Cliffs, N.J., 1988.

[101] Y.S. Huang and C.Y. Suen. "Combination of Multiple Classifiers with Measurement Values," *Proc. 2nd Int. Conf. on Document Analysis and Recognition,* pp. 598–601, Tsukuba, Japan, 1993.

[102] L.K. Hansen and P. Salamon. "Neural Network Ensembles," *IEEE Trans. Pa t. Anal. Machine Intell.*, Vol. 12, No.10, pp. 993–1001, 1990.

[103] Y.S. Huang, K. Liu, and C.Y. Suen. "The Combination of Multiple Classifiers by A Neural Network Approach," *International Journal of Pattern Recognition and Artificial Intelligence*, accepted for publication.

[104] Y.S. Huang, K. Liu, and C.Y. Suen. "A Neural Network Approach for Multi-Classifier Recognition Systems," *The Fourth International Workshop on Frontiers in Handwritten Recognition*, accepted for publication, 1994.

[105] Y.S. Huang and C.Y. Suen. "A Method of Combining Multiple Classifiers - A Neural Network Approach," *Pattern Recognition and Neural Network of 12th International Conf. on Pattern Recognition*, pp. 473–475, 1994.

[106] L. Lam, Y.S. Huang, and C.Y. Suen. "Combination of Multiple Classifier Decisions for Optical Character Recognition," in C.H. Chen, L.F. Pau and P.S.P. Wang, (eds) *Optical Character Recognition and Document Image Analysis*, under preparation.

[107] D.S. Broomhead and D. Lowe. "Multivariable Functional Interpolation and Adaptive Networks," *Complex Systems*, Vol. 2, pp. 321–355, 1988.

[108] T.J. Moody and C.J. Darken. "Faster Learning in Networks of Locally Tuned Processing Units," *Neural Computation*, Vol. 1, pp. 151–160, 1989.

[109] G.E. Hinton. "Connectionist Learning Procedures," *Artificial Intelligence*, Vol. 40, pp. 185–234, 1989.

[110] A.S. Weigend, D.E. Rumelhart, and B.A. Huberman. "Generalization be Weight-Elimination Applied to Currency Exchange Rate Prediction," *Proc. Int. Joint. Conf. on Neural Networks*, Vol. 1, PP. 837–841, Seattle, 1991.

[111] L.T. Tu, W.W. Lin, Y.K. Chan, and I.S. Shyu. "A PC Based Handwritten Chinese Character Recognition System," *Pre-Proc. International Workshop on Frontiers in Handwriting Recognition*, pp. 349–354, Buffalo, New York, USA, 1993.

[112] R.W. Weeks. "Rotating Raster Character Recognition System," *AIEE Trans. Communications and Electronics*, Vol. 80, pp. 353–359, 1961.

[113] K.I. Maeda, Y. Kurosawa, H. Asada, K. Sakai, and S. Watanabe. "Handprinted *Kanji* Recognition by Pattern Matching Method," *Proc. 6th ICPR*, pp. 789–792, 1982.

[114] C.Y. Suen, J. Guo, and Z.C. Li. "Computer and Human Recognition of Handprinted Characters by Parts," *Proc. of 2nd Int. Workshop on Frontier in Handwriting Recognition*, pp. 161–174, Chateau de Bonas, 1991.

[115] C.Y. Suen, J. Guo, and Z.C. Li. "Analysis and Recognition of Alphanumeric Handprints by Parts," *Proc. of 11th Int. Conf. on Pattern Recognition*, pp. 338–341, Hague, 1992.

[116] G. Srikantan. "Gradient Representation for Handwritten Character Recognition," *Pre-Proc. International Workshop on Frontiers in Handwriting Recognition*, pp. 318-323, Buffalo, New York, USA, 1993.

[117] K. Liu, Y.Q. Chang, and J.Y. Yang. "Algebraic Feature Extraction For Image Recognition Based On An Optimal Discriminant Criterion," *Pattern Recognition*, Vol. 26, pp. 903-911, 1993.

[118] K. Liu, Y.S. Huang, and C.Y. Suen. "Image Classification by Classifier Combining Technique," *SPIE, Neural and Stochastic Methods in Image and Signal Processing III,* Vol. 2304, pp. 210-217.

[119] K. Liu, Y.S. Huang, and et al. C.Y. Suen. "The Discriminant Performance of the Algebraic Feature of Handwritten Character Images," *Pattern Recognition and Neural Network of 12th International Conf. on Pattern Recognition,* pp. 426-428, 1994.

[120] K. Liu, Y.J. Liu, F. Jallut, Y.Q. Cheng, and J.Y. Yang. "Automatic Recognition Of Human Face Images," *ADVANCES IN MODELLING AND ANALYSIS (B),* Vol. 28, pp. 51-57, 1993.

[121] S.W. Lee, J. S. Park, and Y.Y. Tang. "Performance Evaluation of Nonlinear Shape Normalization Methods for the Recognition of Large-Set Handwritten Characters," *Proc. Second International Conference on Document Analysis and Recognition,* pp. 402-407, Tsukuba, Japan, 1993.

[122] K. Liu, Y.S. Huang, and C.Y. Suen. "Optimal Matrix Transform for the Extraction of Algebraic Features from Images," *International Journal of Pattern Recognition and Artificial Intelligence*, Submitted on April 14, 1994.

[123] Y.S. Huang, K. Liu, and C.Y. Suen. "A New Method of Optimizing Prototypes for Nearest Neighbor Classifiers Using a Multi-Layer Network," *Pattern Recognition Letters*, accepted for publication.

[124] Y.S. Huang, K. Liu, and C.Y. Suen. "A New Prototype Optimization Method Based on Multi-Layer Network," *Pattern Recognition*, submitted on July 20, 1994.

[125] K. Fukunaga and P.M. Narendra. ". A Branch And Bound Algorithm for Computing K-nearest Neighbors," *IEEE Trans. Comput.*, Vol. 24, pp. 750–753, 1975.

[126] T.P. Yunck. " A Technique to Identify Nearest Neighbors," *IEEE Trans. Systems Man Cybernet.*, Vol. 6, pp. 678–683, 1976.

[127] E. Vidal. "An Algorithm for Finding Nearest Neighbours in (approximately) Constant Average Time," *Pattern Recognition Letters*, Vol. 4, No. 3, pp. 145–157, 1986.

[128] C.L. Chang. "Finding Prototypes for Nearest Neighbor Classifiers," *IEEE Trans. Comput.*, Vol. 23, pp. 1179–1184, 1974.

[129] G.L. Ritter, H.B. Woodruff S.R. Lowry, and T.L. Isenhour. "An Algorithm For A Selective Nearest Neighbor Decision Rule," *IEEE Trans. Inform. Theory*, Vol. 21, pp. 665–669, 1975.

[130] D. White and G. Hanson. "Optimizing Neural Networks Using Faster, More Accurate Genetic Search," *Proc. 3rd Conf. on Genetic Algor.*, pp. 391–396, Arlington, 1989.

[131] D. Goldberg. Genetic Algorithm in Search, Optimization and Machine Learning, Addison-Wesley, Reading, Mass., 1989.

[132] K. Rose, E. Gurewitz, and G. Fox. "A deterministic Annealing Approach to Clustering," *Pattern Recogtion Letters*, Vol. 11, No. 9, pp. 589–594, 1990. .

[133] G. Qiu, M.R. Varley, and T.J. Terrell. "Improved Clustering Using Deterministic Annealing With a Gradient Descent Technique," *Pattern Recognition Letters*, Vol. 15, No. 6, pp. 607–610, 1994.

[134] L.N. Kanal. "On Pattern, Categories, and Alternative Realities," *Pattern Recognition Letters*, Vol. 14, No. 3, pp. 241–255, 1993.

[135] J. Paik, S. Jung, and Y. Lee. "Multiple Combined Recognition System For Automatic Processing of Credit Card Slip Applications," *Proc. Second International Conference on Document Analysis and Recognition*, pp. 520–523, Tsukuba, Japan, 1993.

[136] T. Matsui, T. Noumi, I. Yamashita, T. Wakahara, and Y. Yoshimuro. "State of the Art of Handwritten Numeral Recognition in Japan – The Results of the First IPTP Character Recognition Competition," *Proc. Second International Conference on Document Analysis and Recognition,* pp. 391–396, Tsukuba, Japan, 1993.

[137] R. Fenrich and J. Hull. "Concerns in Creation of Image databases," *Pre-Proc. International Workshop on Frontiers in Handwriting Recognition,* pp. 112–121, Buffalo, New York, USA, 1993.

[138] E.M. Kleinberg. "Stochastic Discrimination," *Annals of Mathematics and Artificial Intelligence,* Vol. 1, pp. 207–239, 1990.

[139] E.M. Kleinberg and T.K. Ho. "Pattern Recognition by Stochastic Modeling," *Pre-Proc. International Workshop on Frontiers in Handwriting Recognition,* pp. 175–183, Buffalo, New York, USA, 1993.