# Decision Tree Approach to Pattern Recognition Problems in a Large Character Set

Qing Ren Wang

A Thesis

in

The Department

of

Computer Science

Presented in Partial Fulfillment of the Requirements
for the Degree of Ph.D. in the Faculty of
Engineering and Computer Science in
Concordia University
Montréal, Québec, Canada

March 1984

# ABSTRACT

## DECISION TREE APPROACH TO PATTERN RECOGNITION PROBLEMS IN A LARGE CHARACTER SET

Qing Ren Wang, Ph.D.
Concordia University, 1984

Decision tree is a fast classifier in pattern recognition, where a large number of classes can be treated and the decision making time can be minimized by a series of small local decisions. Based on the consideration of entropy reduction, the general tree classifier has been analyzed. Theoretical results show that its search time and error rate are both in the order O(H), and overlap in the order O(H.exp(H)), where H is Shannon's entropy measure of the given problem. The results further reveal that the main difficulties in tree implementation are error accumulation and serious memory requirement due to overlap. Some design principles have been drawn from these behaviors of the decision tree. With entropy reduction over overlap as the objective, a new clustering algorithm, called ISOETRP, has been developed. Overlap is treated in ISOETRP for the first time and practically solved by the overlap table itself. Experimental results show that this clustering algorithm is very powerful in the design of the tree classifier. Some works have also been done on feature analysis. The profile

feature and SVD (Singular Value Decomposition) are compared. The phase feature has been proposed and analyzed for pattern recognition. In order to enhance the tree classifier, some branch-bound search algorithms with fuzzy membership function as heuristic evaluation have been developed to reduce the error rate. A new model of decision tree with global training has been proposed. The advantage of the new model over the conventional decision tree is that error accumulation has been suppressed considerably and a very low error rate can be obtained at high speed. Several tree classifiers have been implemented to recognize 500-3200 Chinese characters. The result of the 3200 character classifier is very encouraging : the recognition rate is 99.93%, the error rate only 0.025%, and the speed 861 samples per second when the program is written in Pascal and run on a CYBER-172 computer. These results confirm the theory developed and design principles given in this thesis, as well as the newly proposed decision tree model.

## ACKNOWLEDGEMENTS

I wish to thank Dr. C. Y. Suen, who encouraged me to join this program and agreed to be my supervisor. His advice and counsel during the entire preparation of this thesis are deeply appreciated.

I feel grateful to Drs. R. De Mori, R. Shinghal, S. Lin, V. Ramachandran and A. Krzyzak, Dean M. N. S. Swamy and Professor K. S. Fu, for their useful discussions and suggestions. I take special pleasure in thanking those people in the Dept. of Computer Science and Computer Center of Concordia University, who provided me with the most stimulating environment to pursue my research. I also gratefully acknoowledge all the sacrifices and patience accorded me by my parents, wife and children throughout the period of study.

I also acknowledge the support granted me by the Education Ministry of The People's Republic of China, the support of the Natural Sciences and Engineering Research Concil of Canada and the Kung Chung Wo Co. Ltd. of Hong Kong, awarded to Dr. C. Y. Suen.

TABLE OF CONTENTS

## LIST OF FIGURES

LIST OF TABLES

## LIST OF IMPORTANT ABBREVIATIONS AND SYMBOLS

$C_k$ -- The kth class in the character set.

$S(.,.)$ -- Similarity measure.

$\|.\|$ -- Norm in a Banach space.

K-L -- Karhunan-Loeve transform.

$\lambda_j$ -- The jth eigen velue in K-L expansion.

$\Phi_j$ -- The jth eigen vector.

$\omega_1, \ldots, \omega_n$ -- n pattern classes.

$\Omega$ -- Pattern space.

$\underline{X}$ -- Pattern vector.

$P_i$ -- a-priori probability of $\omega_i$.

$p(\underline{X}|\omega_i)$ -- Probability density function of $\omega_i$.

$p(\omega_i|\underline{X})$ -- Posteriori probability density of $\omega_i$.

$\lambda(\omega_j|\omega_i)$ -- Loss function in decision making.

$\lambda_r$ -- Rejection threshold.

$l^j(\underline{X})$ -- Risk at vector X.

$\omega^*$ -- Bayes' decision.

$L^*$ -- Bayes' risk.

$E^*$ -- Bayes' error rate.

$\nu_i$ -- Mean vector of class $\omega_i$.

$\Sigma_i$ -- Covariance matrix of class $\omega_i$.

$\Omega_i$ -- The region for class $\omega_i$.

$\Phi$ -- Empty set.

$H_n$ -- Shannon's entropy of n probabilities.

$\bar{T}$ -- Average search time.

$\Delta H$, $\Delta \tilde{H}$ -- Entropy reduction at one level.

$H_0$ -- Minimum entropy reduction at each internal node.

OP -- Overlap + 1.

G -- Gain.

$G_0$ -- Minimum gain.

e -- Error rate.

$e_0$ -- Minimum error rate committed at each internal node.

ENTRP -- Entropy reduction in clustering.

OLAP -- Overlap in clustering.

BLNC -- Balance factor in clustering.

GAIN -- Gain in clustering.

PIDX -- Performance index in ISODATA or KMEANS.

$\sigma_i$ -- Deviation within the ith class.

$\sigma$ -- Overall deviation.

$\lambda_1, \ldots, \lambda_r, \mu_1, \ldots, \mu_r$ -- Eigen values.

$\delta_{ij}$ -- Kronecker $\delta$.

diag(...) -- Diagonal matrix.

Trace(.) -- Matrix trace.

Im(.), Re(.) --Imaginary and real parts of a complex number.

$A_{ij}$ -- The original pattern matrix.

$\theta_{ij}$, $M_{ij}$ -- Phase and magnitude of Fourier transform.

$d_{.,.}$ --Distance between two points along a feature line.

$x^i$ -- Candidate center in phase feature space.

$D^i$ -- Candidate deviation of phase feature.

MV -- Membership value.

N(.,.) Noise matrix.

R(i,j) Two dimensional discrete correlation function.

E(.) Mathematical expectation.

D(.) Variance.

# CHAPTER ONE

# INTRODUCTION:

# LARGE CHARACTER SET AND TREE CLASSIFIERS

## 1.1. Chinese Script -- A Large Character Set

Character sets in most languages have become fully phoneticized and alphabetized after the pictographic stage. Owing to this reason, every character set in these languages is usually small. For example, there are only 26 letters in English , 32 in Russian, 24 in Greek, etc. Chinese script may be the only one, which remains logographic, among the popular languages in the world. In the long history, the total number of Chinese characters became larger and larger. There are about 70000 characters in the largest contemporary Chinese dictionary (Table 1.1 and ref.[1.1]).

In Chinese, each character denotes a word or morpheme and corresponds to a syllable with a tone. Characters are written in 1 to 36 standard strokes which are respected in various combinations (Fig.1.1). Traditionally there are 6 categories in the Chinese character set, but 3 of which mainly remain now :

A) Pictograms, e.g.

" 木 " (wood or tree),

"山" (mountain),

"上" (upward),

"下" (downward).

B) Ideographic compounds, e.g.

"休" (a man sitting under a tree, meanng to rest),

"明" (the sun and the moon together, meaning bright).

C) Phonetic compounds, which may consist of 2 parts, one being a radical with simple morpheme, the other a phoneme to indicate the exact or approximate pronunciation of the character. For example, the character "妈" (mother) has the radical "女" (woman) and its pronunciation is /ma/ which is derived from its right part "马".

The percentages of these categories are listed in Table 1.1.

Nowadays in China, about 3000—6000 characters are used in newspapers. Even pupils have to learn 3000—4000 characters in the primary school. About 7000—8000 characters are collected in small dictionaries. The Chinese character set may be the most difficult script for human to write and recognize. Since 1956, work on the graphic simplification of the characters has been carried out in the

TABLE 1. 1 Chinese Characters as Logographics

| Periods | Pictograms | Ideographic Compounds | Phonetic Compounds | Total No. |
|---|---|---|---|---|
| Late Shang 1300 B.C. | 22.59% | 32.30% | 37.76% | 1226 |
| Eastern Han A.D. 100 | ~ | ~ | 80.00% | 9353 |
| Northern Song A.D. 1100 | 3% | 7% | 90.00% | 24253 |
| Present* | ~ | ~ | >90.00% | 70000 |

* Estimation

People's Republic of China, and 2106 frequently used characters have been simplified. Some examples are shown in Fig.1.2, which are much simpler than before. But Chinese characters are still difficult to recognize both by human and by computer for the following reasons:

A) The number of strokes in each character can be very large, e.g. the character "齉" has 36 strokes (pronounced as /nang/ and means snuffle with a cold, or speaking through the nose).

B) The number of characters is very large. As stated above, 3000--6000 characters are often used nowadays and 7000--8000 characters are collected in small dictionaries.

C) Many characters look alike, e.g. between "候" (weather) and "侯" (a nobleman or a high officer), there is only a small difference of a short stroke " | " in the middle.

D) There are some character pairs which have exactly the same shape but oriented in different directions, e.g. "四" (four) and "目" (eye), "人" (person) and "入" (enter), "由" (from) and "甲" (first rate), etc.

The Japanese script is another example of a large character set. In Japan, people use 50 hiraganas, 50 katakanas (Japanese letters) and about 2000 Kanji's

Fig. 1.1 Examples of Chinese Characters


Fig. 1.2 Complex and Simplified Characters

| | | |
|---|---|---|
| 才 纔 | 蚕 蠶 | 忏 懺 |
| 谷 穀 | 驴 驢 | 痈 癰 |
| 胡 鬍 | 毡 氈 | 医 醫 |
| 厅 廳 | 吁 籲 | 丛 叢 |

(Chinese characters).

To recognize such a large set of characters, Bayes' classifier is practically impossible. Due to the large number of decision regions present in the feature space, some conventional classifiers, such as linear or K-NN (the K-Nearest Neighbors) [1.2] would be very inefficient both in time and storage, if not totally impossible. Only 2 types of classifiers have been reported to recognize such large character sets: one is matching scheme [1.4, 1.5, 1.24], the other is a decision tree originated from this thesis. Both types will be introduced in Sections 1.2 and 1.3 respectively.

## 1.2. Matching Scheme

Chinese character recognition research began in the mid 60's [1.3] and first success was reported in the late 70's [1.4, 1.5, 1.24]. Most developments in this period were based on the matching scheme, or similarity method described below.

Suppose an unknown pattern X is to be classified as a member of one of the classes in the candidate set

$$\{C_k \mid k = 1, 2, \ldots, n\}.$$

The similarity between X and each $C_k$, $S(X,C_k)$ is calculated and the category $C_{k_0}$ with the largest $S(X,C_k)$ value among $C_k$'s is assigned to the unknown pattern.

The simple similarity measure can be defined as

$$S(X,C_k) = (X, C_k) / ((X, X)(C_k, C_k))^{1/2}$$

$$k = 1, 2, \ldots, n \qquad (1.1)$$

where $(X,X')$ is the inner product properly defined for the patterns X and X'.

A very powerful similarity measure, multiple similarity was developed in [1.6], which is essentially an application of Karhunan-Loève expansion. Let us describe the result in [1.6] for explanation and later referencing. Suppose the

unknown pattern X and a candidate $f=C_k$ are given, the latter having the sample set $\{f_a | a \in D\}$. D is the index space, and the probability of index a is P(a). The multiple similarity between X and f is defined by

$$S^2(X,f) = \int_D p(a)(X,f_a)^2 / (\|X\|^2\|f_a\|^2)\, da \qquad (1.2)$$

Suppose each pattern $f_a$ (or X) is a function in a Banach space $\underline{R}$. In picture or optical character recognition (OCR), these functions are the light energy distributed over a plane region $r \in \underline{R}$. The kernal function of the category f is

$$K(r, r') = \int_D p(a)\, f_a(r)\, f_a(r')\, da \qquad (1.3)$$

According to Karhunan-Loève expansion

$$K(r, r') = \sum_{j=1}^{\infty} \lambda_j \Phi_j(r) \Phi_j(r') \qquad (1.4)$$

where $\lambda_j$'s and $\Phi_j$'s are the eigen values and eigen functions of the following integral equation respectively,

$$\int_{\underline{R}} K(r, r') \Phi(r')\, dr' = \lambda_j \Phi_j(r) \qquad (1.5)$$

and

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m \geq 0.$$

By (1.4) the multiple similarity (1.2) can be expressed as

$$S^2(X, f) = \sum_{j=1}^{\infty} \frac{\lambda_j}{\lambda_1} (\phi_j, X)^2 / \|X\|^2 \qquad (1.6)$$

Often, only a few terms of the right hand side are used to calculate the multiple similarity. In simple similarity, usually only

$$\Phi(r) = \int_D p(a) f_a(r) da \qquad (1.7)$$

is used to express $C_k$ in (1.1). The multiple similarity is much better than the simple similarity given in (1.1), but the inter-relation between the candidates is not considered in any case. When 2 candidates are close to each other, the misclassification rate can be high. An improved version of (1.6), the compound similarity has been developed [1.4] to solve this problem.

In a "pure" matching scheme, formula (1.1) or (1.6) is computed n times to classify an unknown pattern into one of n given candidates, as explained in Fig.1.3. When n is very large, e.g. 3000—6000 or larger in Chinese character recognition, this becomes very inefficient.

In the work reported in [1.5], a two stage classifier (Fig.1.4) was actually used to classify about 2000 characters. At the first stage, the unknown pattern is classified into one of about 20 groups. At the second stage, this unknown is further compared with about 100 candidate characters within this group and then assigned to

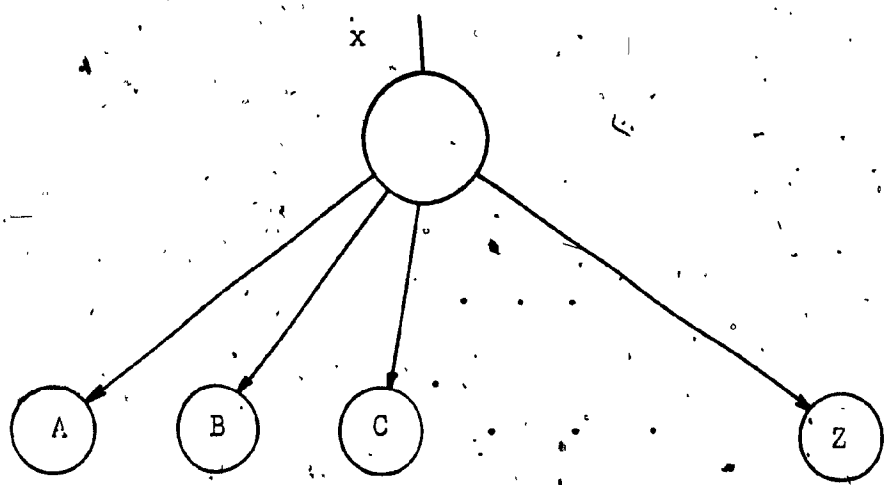



Fig. 1.3 Pure matching scheme

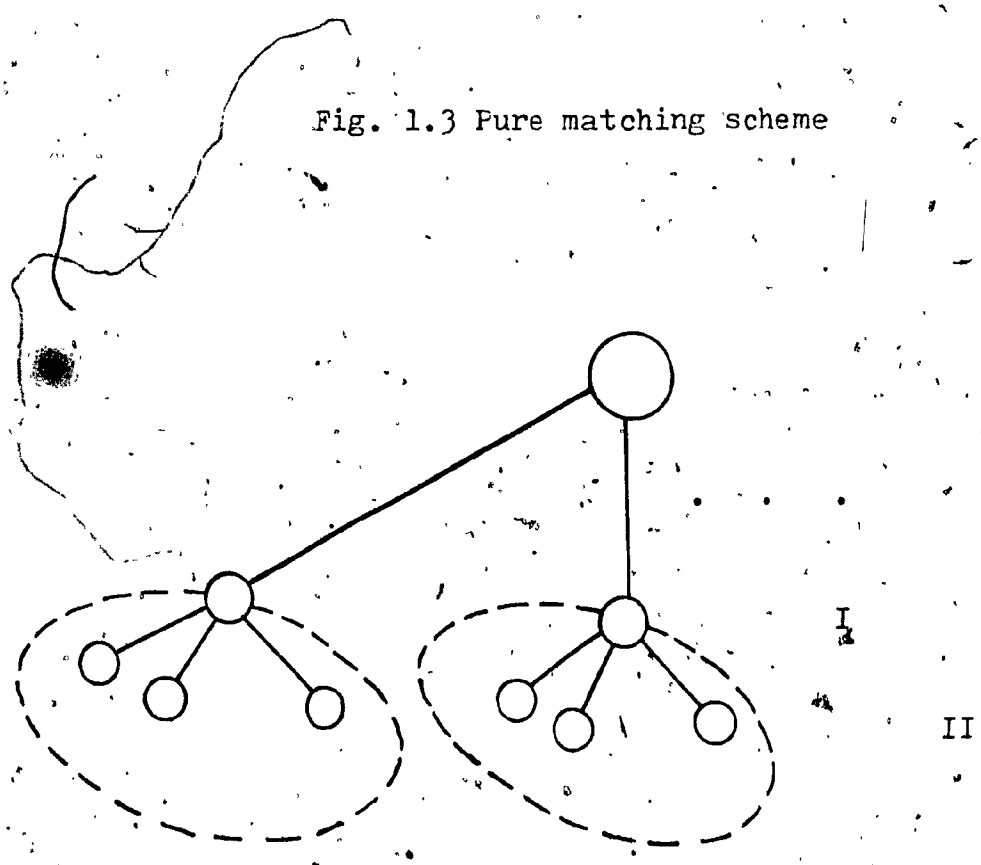



Fig. 1.4 Two stage classifier

a character category. One can see, the time requirement of a best two stage classifier is still in the order $O(n^{1/2})$ and usually another group of features other than $\Phi_j$'s is needed by the classification process at the first stage. To speed up the classifier, hardware was actually employed in [1.5] and its previous versions.

## 1.3. Tree Classifiers

The decision tree classifier has been widely used in various pattern recognition problems, such as geoscience and remote sensing [1.8, 1.9], speech analysis [1.10], biomedical problems [1.11, 1.12, 1.13]. Some analysis and design methods are also given in [1.20—23]. In a tree classifier, the global decision task can be accomplished via a series of local decisions. The tree classifier makes use of different features at different stages. This property enables it to perform well in the following two main types of applications.

1) The decision region is complex and usually in a high dimensional space, but the number of classes is small.

The local decision region at each level of the tree classifier is a simple one in a lower dimensional space. The global decision region can be approximated very well by combining these local regions at various levels. The applications of the tree classifier mentioned earlier belong to this type.

An example decision tree from [1.12], which differentiates cancer cells from normal cells is shown in Fig.1.5. The cells are subdivided into n=4 classes, and one feature is used at each internal node of the tree. Each terminal node belongs to one of the four

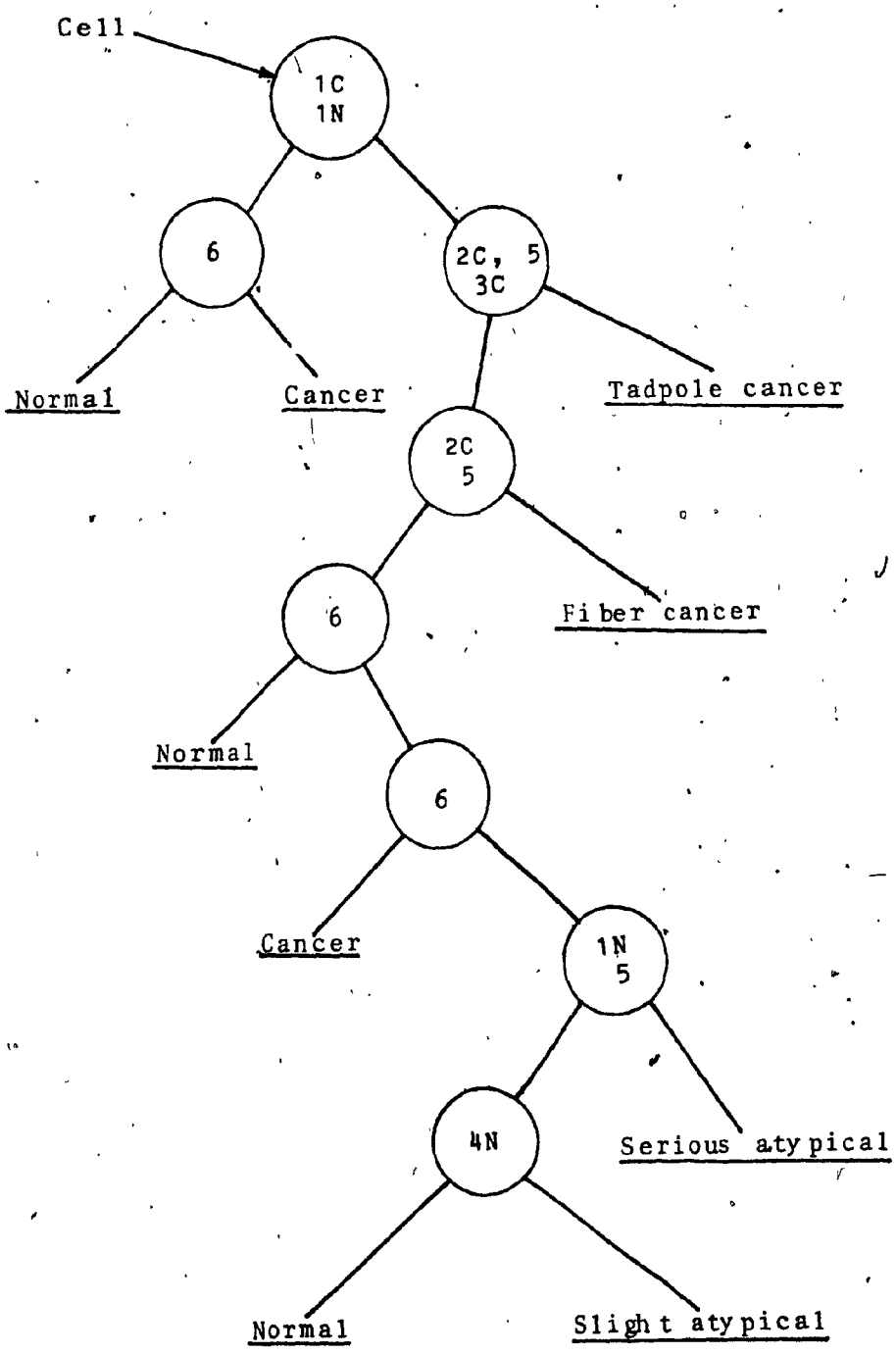



Fig. 1.5 A binary tree application

classes.

In the above applications, analysis and design methods, usually the binary decision tree was presumed. As reported before, the problems to be considered in the design of a tree are balance factor, overlap (Chapter 2) and feature selection, etc. Since n is not very large in the above works, the difficulties encountered in large character set problems were not reflected.

2) The number of classes , n, in the given problem is very large, e.g. $N \geq 3000$ in the Chinese character recognition [1.14].

A large number of classes can be handled in the decision tree. Compared to the single stage classifier, where n classes are separated in one step, the tree classifier makes it possible to reduce the decision making time significantly. For example, in a complete, balanced binary tree with n terminals, the time requirement is $O(\log_2 n)$ [1.7]. While in a single stage classifier, the time requirement in an n class problem is in $O(n)$. Even in an optimized two stage classifier, the decision making time, $O((n)^{1/2})$, is still far larger than $O(\log_2 n)$.

The application of the decision tree classifier to Chinese character recognition was proposed by the author and simulated in [1.14, 1.15, 1.16, 1.17, 1.18], where n was

very large (between 500 and 3200). An example from reference [1.14] is shown in Fig.1.6. In this work, 64 Walsh transforms of some profile features (see Section 4.2) were used to classify 3155 noisily printed characters. The tree is a general one, i.e. the number of branches issued from each internal node is not presumed to be 2. Only one feature was used at each internal node, which is indicated by its identity number in each round circle. Actually only a simple inequality test is needed in the recognition phase, resulting in a very high speed. The character in each terminal node is written in a square frame. Both internal nodes and terminal nodes have the same structure type in the computer storage, which is as follows when using Pascal declaration.

```
TYPE
    LNK = ^NODE;
    NODE = record
            FN : integer;
                { denoting feature number in an internal
                node or a character i.d. in a terminal
                node }.
            FEA : real;
                { some constant in an inequality }
            BELOW : LNK;
                {  to the first child node of the
                internal node, or nil in a terminal node
```

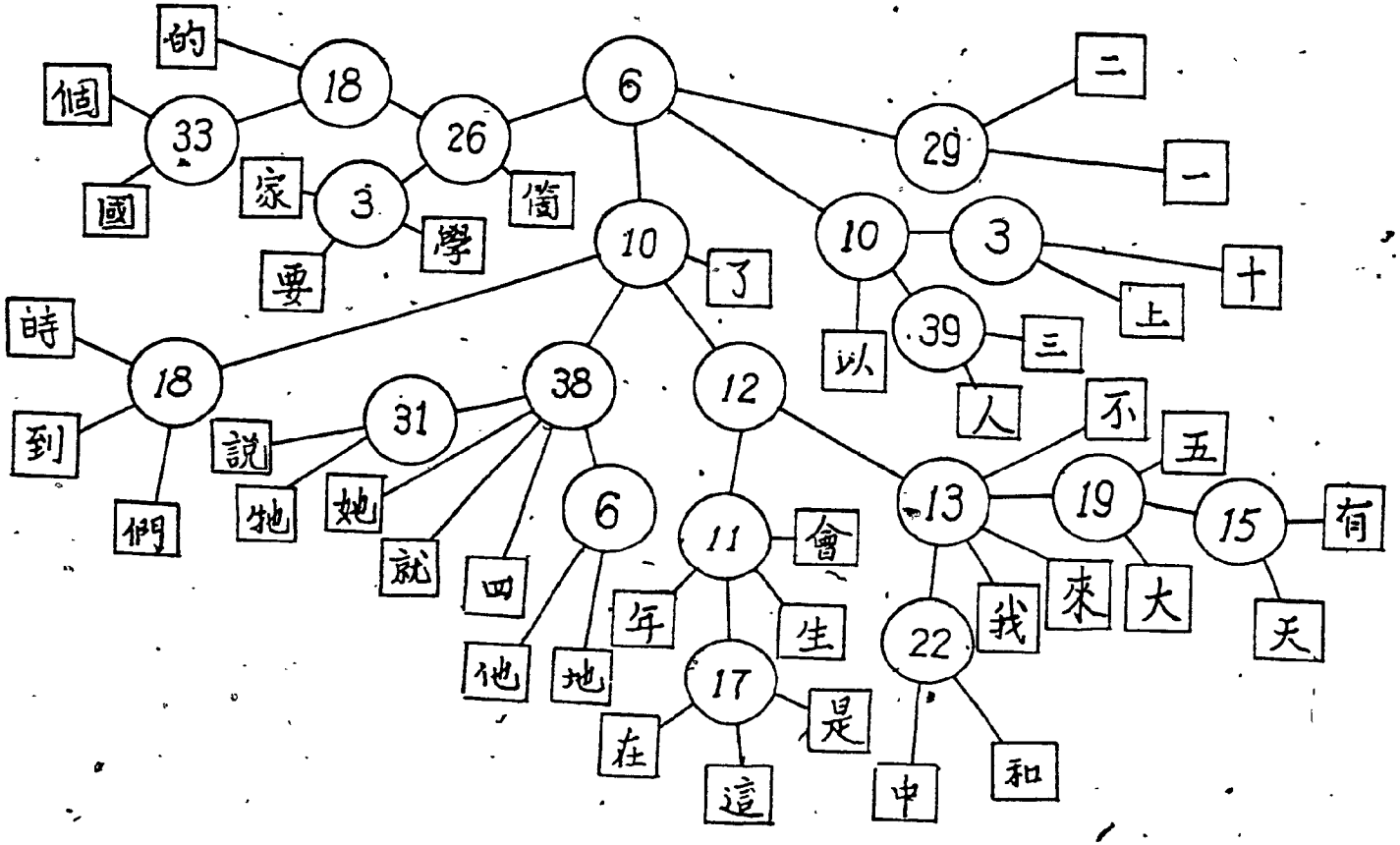



FIG. 1.6 An Example of Decision Tree for 40 Characters

```
        }
    NEXT : LNK
        { to the next sibling node }
end;
```

Using this structure, the general tree is actually mapped to a binary tree in memory. The binary tree corresponding to Fig.1.6 is shown in Fig.1.7. In this way, the general tree gives more flexibility than the binary tree without taking up more memory space.

In this example, only one feature was used at each stage and the decision region of each character consists of one or more hyper rectangulars in the 64-dimensional space. A more sophisticated classifier can be obtained if we use more than one feature in each internal node. A tree example which makes use of 4 features at each internal node is shown in Fig.1.8 [1.15, 1.16, 1.17, 1.18]. For these trees, the above data structure is modified: FN becomes an array of 4 integers, which represent the feature numbers used in this internal node; FEA is changed to an array of 4 real numbers, storing the center coordinates of the character subgroup corresponding to this node (internal or terminal). The local decision is made by comparing the distances between the unknown pattern and these centers in this 4-dimensional space. In this case, the tree is essentially a piecewise linear classifier.

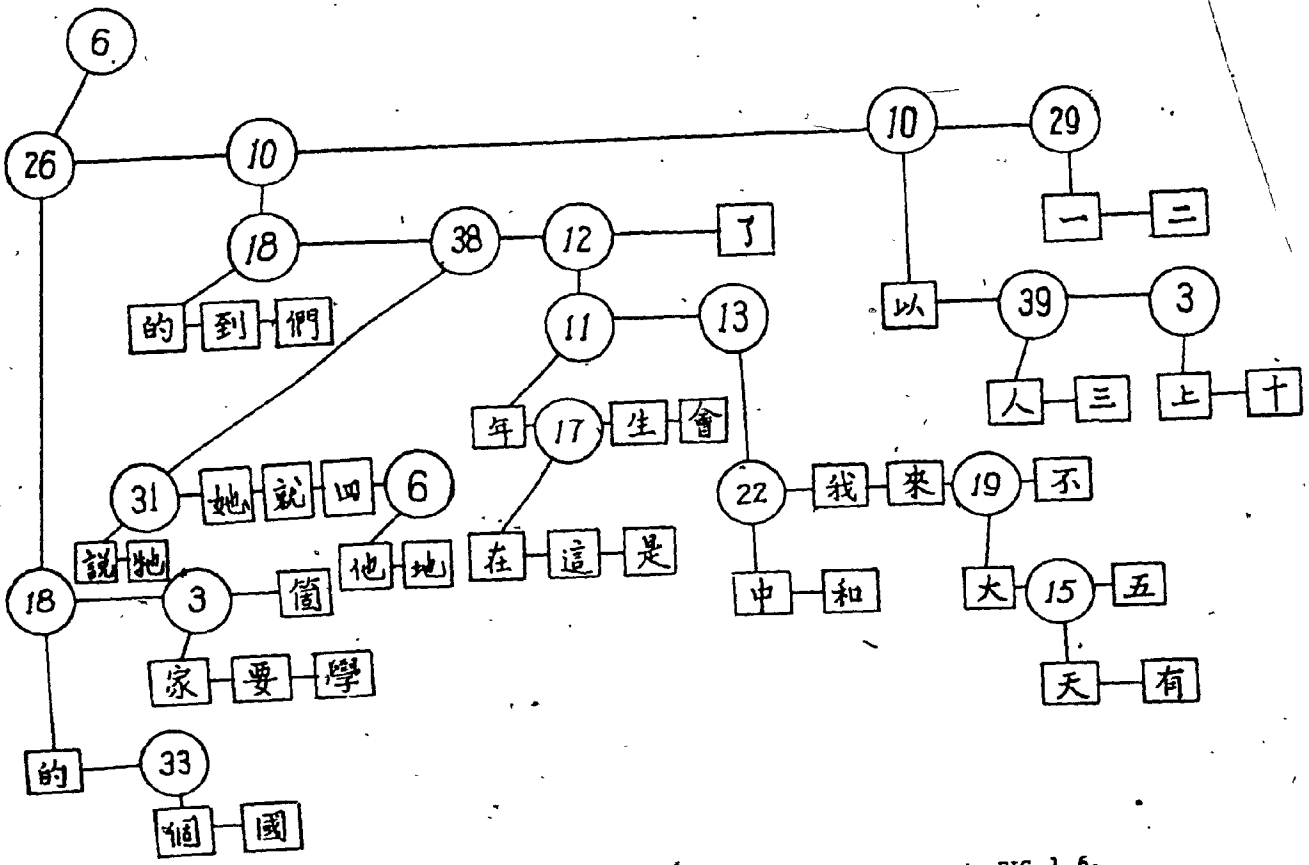



FIG. 1.7 The Equivalent Binary Tree of the Decision Tree in FIG. 1.6.

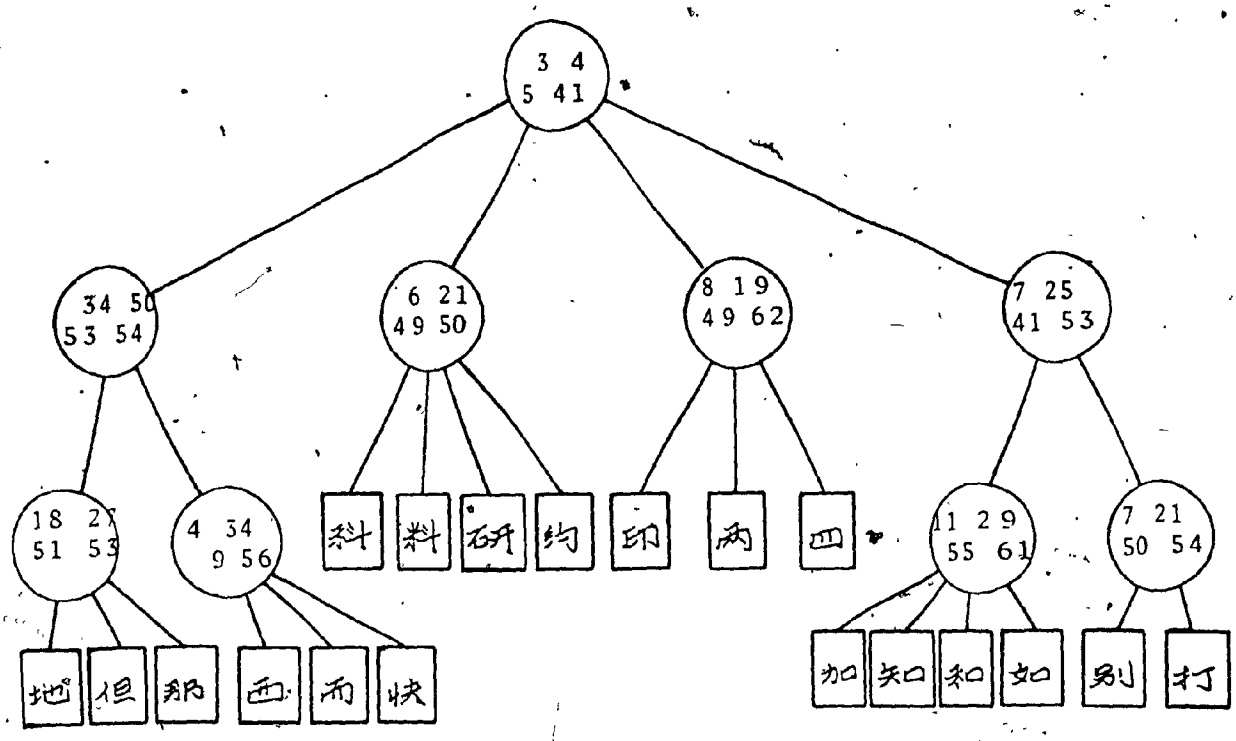



Fig.1.8 A general tree with 4 features used in each internal node

From the data structure of the tree classifier, one can see it is very important to use uniform features and descriminants in the classification process. Otherwise the tree would take too much memory and make the classifier practically impossible in a large character set problem.

## 1.4. Main Parts of the Thesis

When the decision tree is applied to pattern recognition in large character sets to raise the time efficiency, many difficulties in the implementation are encountered which must be overcome. Usually the storage requirement has a higher order than O(n), n being the number of classes in the given problem, due to overlap in the tree. Owing to error accumulation in various levels of the tree, the total error rate is larger than that produced by a matching scheme if no special treatment is applied. Both feature extraction and selection in matching scheme can be solved by Karhunan-Loève expansion easily (although not fast). While in the decision tree, feature extraction and selection are usually different from K-L expansion, which is explained in Section 4.1.

To overcome these difficulties, the decision tree is first analyzed based on entropy reduction, which is described in Chapter 2. Some design principles are then drawn from the analysis, and solutions are proposed according to these principles in the ensuing chapters. In Chapter 3, a new clustering algorithm ISOETRP with new objectives is proposed. These objectives differ from those of any traditional clustering analysis. They serve the design of tree classifiers very well. An interactive version of ISOETRP has been developed, which solves the overlap problem by means of an "overlap table". It seems to be very powerful and very effective in practice.

Chapter 4 analyzes why K-L transform does not fit the tree classifier. Some analyses about feature extraction (profile, SVD and phase features) and merit measure are also included in this Chapter. Chapter 5 is devoted to the methods of reducing the error rate of the tree classifier, where some branch-bound search algorithms, with fuzzy membership function as the heuristic evaluation, are proposed. A new concept of global training is proposed, which compensates the error accumulation (or local behavior) of the tree classifier. After that a new model tree classifier is actually proposed. Experimental results show that it gives a much better performance than the old model with only straight forward search.

In Chapter 6, all simulation experiments on tree classifiers in both old model and new model are summarized. The descriptions and recognition results of these trees justify the analysis of the tree classifier, the design principles and the ISOETRP clustering algorithm, and show that the newly proposed tree classifier model is a good approach to the large character set problem.

Conclusions of this thesis are given in Chapter 7.

## 1.5. References


[1]. P. T. Ho, The Cradle of the East, 254-258, Hong Kong University Press, 1972.

[2]. K. Fukunaga, Introduction to Statistical Pattern Recognition, Academic Press, New York, 1972.

[3]. R. Casey and G. Nagy, "Recognition of printed Chinese characters," IEEE Trans. Elec. Comput., vol.15, 91-101, 1966.

[4]. K. Mori and I. Masuda, "Advances in recognition of Chinese characters," Proc. 5th Int. Conf. Pattern Recognition, 692-702, Dec. 1980.

[5]. S. Hirai and K. Sakai, "Development of a high performance Chinese character reader," Ibid., 867-871.

[6]. T. Iijima, "A theory of character recognition by pattern matching method," Proc. 1st Int. Joint Conf. Pattern Recognition, 50-56, 1973.

[7]. A. V. Aho, J. E. Hopcroft and J. D. Ullman, The Design and Analysis of Computer Algorithms, Addison-Wesley, Don Mills, Ont., 1974.

[8]. P. H. Swain and H. Hauska, "The decision tree classifier: design and potential," IEEE Trans. Geoscience Electronics, vol. GE-15, 142-147, July 1977.

[9]. K. C. You anf K. S. Fu, "An approach to the design of a linear binary tree classifier," Proc. 3rd Symposium on Machine Processing of Remotely Sensed Data, Purdue University, pp. 3-A-1-3-A-10, June-July, 1976.

[10]. G. R. Dattatreya and V. V. S. Sarma, "Decision tree design for pattern recognition including feature measurement cost," Proc. 5th Int. Conf. on Pattern Recognition, pp. 1212-1214, Dec. 1980.

[11]. Y. K. Lin and K. S. Fu, "Automatic classification of cervical cells using a binary tree classifier," Ibid., pp. 570-574.

[12]. C. J. Chen and Q. Y. Shi, "Shape features for cancer cell recognition," Ibid., pp. 579-581.

[13]. Q. Y. Shi, "A method for the design of binary tree classifiers," IEEE Conf. on Image Processing and Pattern Recognition, Dallas, pp. 21-26, Aug. 1981.

[14]. Y. X. Gu, Q. R. Wang and C. Y. Suen, "Application of multi-layer decision tree in computer recognition of Chinese characters," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 5, 83-89, Jan. 1983.

[15]. Q. R. Wang, Y. X. Gu and C. Y. Suen, "A preliminary study on computer recognition of Chinese characters printed in different fonts," Proc. Int. Conf. of the Chinese Language Computer Society, 344-351, Sept. 1982.

[16]. Q. R. Wang and C. Y. Suen, "Classification of Chinese characters by phase features and fuzzy logic search," Proc. 1983 Int. Conf. on Chinese Infomation Processing, pp. 133-155 Oct., Beijing.

[17]. Q. R. Wang and C. Y. Suen, "Decision tree with heuristic search and global training, and application to Chinese character recognition," in preparation.

[18]. Y. Y. Tang, C. Y. Suen and Q. R. Wang, "Chinese character classification by globally trained tree classifier and Fourier desciptors of condensed patterns," in press, IEEE 1st Int. Conf. Computers & Applications, Beijing, 1984.

[19]. Frequencies of Chinese Characters, Beijing Xinhua Press, 1976.

[20]. I. K. Sethi and B.Chatterjee, "Efficient decision tree design for discrete variable pattern recognition problems," Pattern Recognition, vol. 9, 197-206, 1977.

[21]. A. V. Kulkarni and L. N. Kand, "An optimization approach to hierarchical classifier design," Proc. 3rd Int Joint Conf. on Pattern Recognition, pp.459-466, Coronado, CA, 1976.

[22]. P. Argentiero, R. Chin and P. Beaudet, "An automated approach to the design of decision tree classifiers," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. PAMI-4, No.1, pp.51-57, 1982.

[23]. I. K. Sethi and G. P. R. Sarvarayuda, "Hierarchical classifier design using mutual information," Ibid. No.4, pp.441-445, 1982.

[24]. C. Y. Suen, "Computer recognition of Kanji characters," Proc. Int. Conf. Text Processing with a Large Character Set, 429-436, Oct. 1983.

# CHAPTER TWO

# ANALYSIS OF THE DECISION TREE CLASSIFIER

## INTRODUCTION

In addition to Bayes decision theory, entropy reduction has been proved useful in the analysis of tree classifiers. It can be shown that the searching time and error rate of the decision tree without overlap are both of order O(H), where H is Shannon's entropy measure of the given problem. This result can also be extended to the overlap case. Furthermore, the theoretical result shows that the memory requirement of the tree classifier with overlap is of order O(H.exp(H)), or in another form, O(n.log n) where n is the number of classes in the given problem. These theorems reveal that the main difficulties in implementing large tree classifier are excess memory requirement caused by overlap and error accumulation. To solve these as well as other difficulties, some design principles are proposed based on the above theorems.

## 2.1. Bayes Decision Theory and Entropy Reduction

Statistical methods have played a major role in pattern recognition developments [2.1-2.3]. Even in some other successful approaches such as syntactic methods, statistical tools may also be useful [2.5, Chap.6]. In this section, we introduce the following definitions, notations and theoretical results (up to eqn. 2.9), which are similar to those given in [2.4, Chap.2].

Suppose there are n possible pattern classes $\omega_1, \omega_2, \ldots, \omega_n$ and an arbitrary pattern belongs to class $\omega_i$ with a priori probability $P_i$,

$$\sum_{i=1}^{n} P_i = 1, \quad P_i > 0.$$

after feature extraction, patterns are expressed as d-dimensional vectors. When pattern $\underline{X}$ is known to belong to class $\omega_i$, it is assumed to be a random vector taking value in a d-dimensional feature space $\underline{R}$ with the multivariate probability density function $p(\underline{X}|\omega_i)$, $i=1,2,\ldots,n$. The probability of $\underline{X}$ belonging to class $\omega_j$ is called a posteriori probability $P(\omega_j|\underline{X})$, which can be computed by Bayes' rule:

$$p(\omega_j|\underline{X}) = P_j \, p(\underline{X}|\omega_j) \; / \; p(\underline{X}), \qquad j=1, 2, \ldots, n \tag{2.1}$$

where

$$p(\underline{X}) = \sum_{i=1}^{n} p(\underline{X}|\omega_i)P_i \qquad (2.2)$$

is the unconditional probability, or overall distribution density of $\underline{X}$.

Let $\hat{\omega}(\underline{X})$ be some decision rule, i.e. the class to which the pattern $\underline{X}$ is assigned. For example, $\hat{\omega}=\omega_i$ means the pattern $\underline{X}$ is assigned to class i, i=1,2,...,n. Furthermore $\hat{\omega}=\omega_0$ means $\underline{X}$ is rejected. Let $\lambda(\omega_j|\omega_i)$ be a measure of loss incurred when $\hat{\omega}(\underline{X})=\omega_j$ is made and the true pattern class $\omega$ is in fact $\omega_i$, i=1, 2, ..., n, j=1,2,...,n. The conditional risk of making decision $\hat{\omega}(\underline{X})=\omega_j$ is

$$l^j(\underline{X}) = \sum_{i=1}^{n} \lambda(\omega_j|\omega_i)p(\omega_i|\underline{X})$$

and the average risk is

$$L = \int_{\Omega} l(\underline{X})d\underline{X} \qquad (2.3)$$

where $\Omega$ is the entire pattern space, and

$$l(\underline{X}) = \sum_{i=1}^{n} \lambda(\hat{\omega}(\underline{X})|\omega_i)p(\omega_i|\underline{X}) \qquad (2.4)$$

The decision rule which has the minimum risk is Bayes' decision rule

$$\hat{\omega}^* = \omega_i, \text{ if } l^i(\underline{X}) \leq l^j(\underline{X})$$

$$\text{for } j = 1, 2, \ldots, n \qquad (2.5)$$

The corresponding risk is Bayes risk

$$L^* = \int_{\omega} l^*(\underline{X})p(\underline{X})d\underline{X} \qquad (2.6)$$

where

$$l^* = \underset{i=1,2,\ldots,n}{\text{MIN}} \; l^i.$$

When the loss function is

$$\lambda(\omega_i|\omega_j) = \begin{cases} 1, & i \neq j,\ i=1,2,\ldots,n,\ j=1,2,\ldots,n \\ 0, & i=j=1,2,\ldots,n \\ \lambda_r, & i=0,\ j=1,2,\ldots,n \end{cases} \qquad (2.7)$$

where $\lambda_r$ is the rejection threshold, Bayes' decision rule becomes

$$\omega^*(\underline{X}) = \begin{cases} \omega_i \text{ if } p(\omega_i|\underline{X}) = p^*(\underline{X}) \geq 1-\lambda_r \\ \omega_0 \text{ if } p_* < 1-\lambda_r \end{cases} \qquad (2.8)$$

when

$$p^*(\underline{X}) = \underset{j=1,2,\ldots,n}{\text{MAX}} \; p(\omega_j|\underline{X}).$$

If $\lambda_r = 0$, Bayes rule becomes simpler and the minimum error rate is

$$E^* = \int_{\Omega}(1-\underset{j}{\text{MAX}}\; p(\omega_j|\underline{X}))p(\underline{X})d\underline{X}. \qquad (2.9)$$

Let us consider an example in which each class has a multivariate normal distribution density

$$p(\underline{X})|\omega_i) =$$

$$(2\pi)^{-m/2}|\Sigma_i|^{-1/2}\exp[-(\underline{X}-\underline{\mu}_i)^t\Sigma_i^{-1}(\underline{X}-\underline{\mu}_i)/2]$$

where $\underline{\mu}_i$ and $\Sigma_i$ are the mean vector and covariance matrix of classi $\omega_i$ respectively, m is the dimension of the feature space and t denotes transposition. Let the regions be

$$\Omega_i = \{\underline{X} | (\underline{X}-\underline{\mu}_i)^t\Sigma_i^{-1}(\underline{X}-\mu_i)<\alpha\},$$

$$i=1,2,...,n \qquad (2.10)$$

where $\alpha$ is a threshold which is as large as possible under the following condition :

$$\Omega_i \cap \Omega_j = \Phi,\ i\neq j,\ \ i=1,2,...,n,\ j=1,2,...,n. \qquad (2.11)$$

Suppose the decision rule

$$\omega(\underline{X}) = \omega_i \text{ when } \underline{X} \in \Omega_i \quad i=1,2,...,n$$

$$\omega_0 \text{ otherwise} \qquad (2.12)$$

gives the error rate E, rejection rate R and correct recognition rate C, then

$$C \geq C_0 = \sum_{i=1}^{n} P_i \cdot \int_{\Omega_i} p(\underline{X}|\omega_i)\, d\underline{x} \qquad (2.13)$$

It is easy to prove that the integration part

$$I = \int_{\Omega_i} p(\underline{X}|\omega_i)d\underline{X} \qquad (2.14)$$

depends only on the space dimension m and the value of $\alpha$. The corresponding formula is deduced in the Appendix of this chapter. By means of this formula, $C_0$ is calculated as the

function of m and $\alpha$, and shown in Table 2.1.

From Table 2.1, we see that, in order to get a high correct recognition rate, a large value of $\alpha$ is preferred. But this is often restricted by the fact that (2.11) no longer holds when $\alpha$ is too large, e.g. $\alpha > 10$ in practice. Let us consider Fig. 2.1 to see how this problem is solved in a tree classifeir. In the d-dimensional space, an $\alpha$ is selected so that the value of C in (2.13) is big enough, such that the integration value I over region $\Omega_i$ in eqn. (2.14) is almost 1. Suppose only 2 features are used in the first stage of the tree, where the projection of the regions $\Omega_i$s corresponding to $\omega_i$s are shown in Fig.2.1a. The pattern classes are divided into 2 groups A and B :

$$A = \{\omega_1, \omega_2, \omega_3, \omega_4\}$$

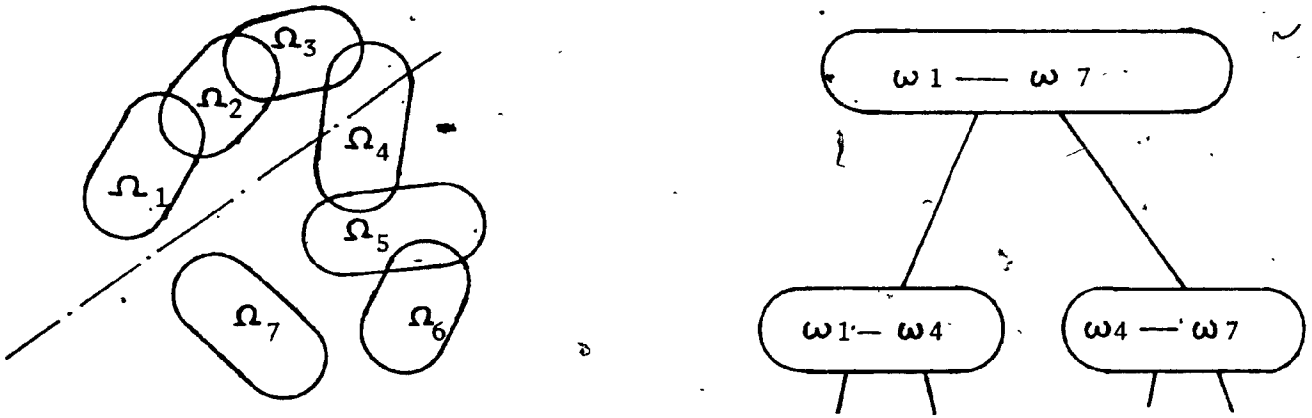
$$B = \{\omega_4, \omega_5, \omega_6, \omega_7\} \qquad (2.15)$$

In each group, the pattern classes can be divided into smaller groups using some other features, forming the next level of the tree. This process continues until only one class remains in each small group, arriving at the terminal nodes of the tree. In this way, a large number of classes can be treated and the decision making time can be minimized by a series of small local decisions.

In any pattern recognition problem, each unknown pattern is assigned to some pattern class. In other words the

Tab. 2.1 The Function $C_0(\alpha,m)$

| alpha | 2.5000 | 2.7500 | 3.0000 | 3.2000 | 3.4000 | 3.6000 | 3.8000 | 4.0000 |
|---|---|---|---|---|---|---|---|---|
| m= 1 | 0.9876 | 0.9940 | 0.9973 | 0.9986 | 0.9993 | 0.9997 | 0.9999 | 0.9999 |
| m= 2 | 0.9561 | 0.9772 | 0.9889 | 0.9940 | 0.9969 | 0.9985 | 0.9993 | 0.9997 |
| m= 3 | 0.9000 | 0.9440 | 0.9707 | 0.9833 | 0.9909 | 0.9953 | 0.9976 | 0.9989 |
| m= 4 | 0.8188 | 0.8910 | 0.9389 | 0.9634 | 0.9791 | 0.9885 | 0.9940 | 0.9970 |
| m= 5 | 0.7174 | 0.8179 | 0.8909 | 0.9313 | 0.9586 | 0.9762 | 0.9870 | 0.9932 |
| m= 6 | 0.6042 | 0.7281 | 0.8264 | 0.8851 | 0.9275 | 0.9563 | 0.9749 | 0.9862 |
| m= 7 | 0.4891 | 0.6272 | 0.7473 | 0.8246 | 0.8840 | 0.9269 | 0.9561 | 0.9749 |
| m= 8 | 0.3807 | 0.5227 | 0.6577 | 0.7514 | 0.8281 | 0.8868 | 0.9290 | 0.9576 |
| m= 9 | 0.2854 | 0.4212 | 0.5627 | 0.6686 | 0.7607 | 0.8356 | 0.8925 | 0.9331 |
| m=10 | 0.2062 | 0.3285 | 0.4679 | 0.5803 | 0.6844 | 0.7741 | 0.8461 | 0.9004 |




a. Overlap due to big $\alpha$

b. The first stage of the tree

Fig. 2.1 Tree classifier solving big $\alpha$ problem

uncerntainty about the class to which the unknown belongs should be removed in the recognition process. This uncertainty can be measured by means of entropy. This idea is especially useful in our decision tree classifier. The entropy is initially large at the root node and it decreases as the unknown pattern goes from an internal node to one of its descending nodes. This entropy decreases level by level until the unknown pattern reaches a terminal node where the entropy vanishes almost completely. We see

$$A \cap B \neq \Phi \tag{2.16}$$

which causes an overlap, i.e., class $\omega_4$ belongs to both groups A and B. If there is no overlap in the decision tree, the problem is as simple as coding the n patterns. Shannon proved [2.8] that the average code length $\bar{L}$ (which is similar to the tree searching time to be described later) satisfies

$$\bar{L} \geq H_n(p_1,p_2,\ldots,p_n)/\log D \tag{2.17}$$

where $H_n$ is Shannon's entropy defined on the probabilities

$$p_1,\ p_2,\ \ldots,\ p_n,$$

and D is the number of symbols used in coding (which is similar to the maximal number of branches in the decision tree), and "log" has a base 2. He also proved that there exists an optimal coding with average code length

$$\bar{L}^* < H_n(p_1,p_2,\ldots,p_n)/\log D+1 \tag{2.18}$$

In many applications, a binary tree is assumed, and balance (i.e. the child nodes have equal size) and completeness of the tree (i.e. all terminal nodes lie at the same level) are emphasized. These conditions often conflict with overlap and do not lead to an optimal tree classifier. In this and the following sections of this chapter, the tree classifier in general form is analyzed from the point of view that entropy is reduced one level at a time in the tree. As pointed out in [2.6, 2.7] and elsewhere, rather than considering pattern recognition problems from Bayes point of view, they can be treated in light of minimizing the entropy, which is defined properly with respect to the given problem.

## 2.2. Decision Tree without Overlap

Let us explain the symbols used in the various sections of this chapter[4] by taking the tree shown in Fig.2.2 as an example. It classifies English letters from A to T (n=20). The root node is at level 0 which contains all the letters

$$\Omega^{(0)} = \{A,B,C,\ldots,T\} \tag{2.19}$$

There are exactly n terminal nodes in this tree, i.e. there is no overlap. We label each terminal node by its corresponding letter. The root node has 3 branches 1, 2 and 3. Let $\Omega_i^{(1)}$ be the entire set of letters in the terminal nodes in the ith branch (i=1,2,3), then

$$\Omega = \cup_{i=1}^{3} \Omega_i^{(1)},$$

$$\Omega_i^{(1)} \cap \Omega_j^{(1)} = \Phi, \quad i \neq j. \tag{2.20}$$

Suppose each letter has a-priori probability $P_j$ (j=1,2,...,n), we define $P_i^{(1)}$ as the probability of $\Omega_i^{(1)}$,

$$P_i^{(1)} = \sum_{j \in \Omega_i^{(1)}} P_j \qquad i=1, 2, 3 \tag{2.21}$$

In general, for a node at level k which has $m_k$ branches, we use the symbol

$$P_i^{(k)} = \sum_{j \in \Omega_i^{(k)}} P_j \qquad i=1, 2, \ldots, m_k \tag{2.22}$$

to denote the probability of the letter set corresponding to its ith child node. If there is no overlap in a subtree rooted at a node at level k-1, then the probability of this

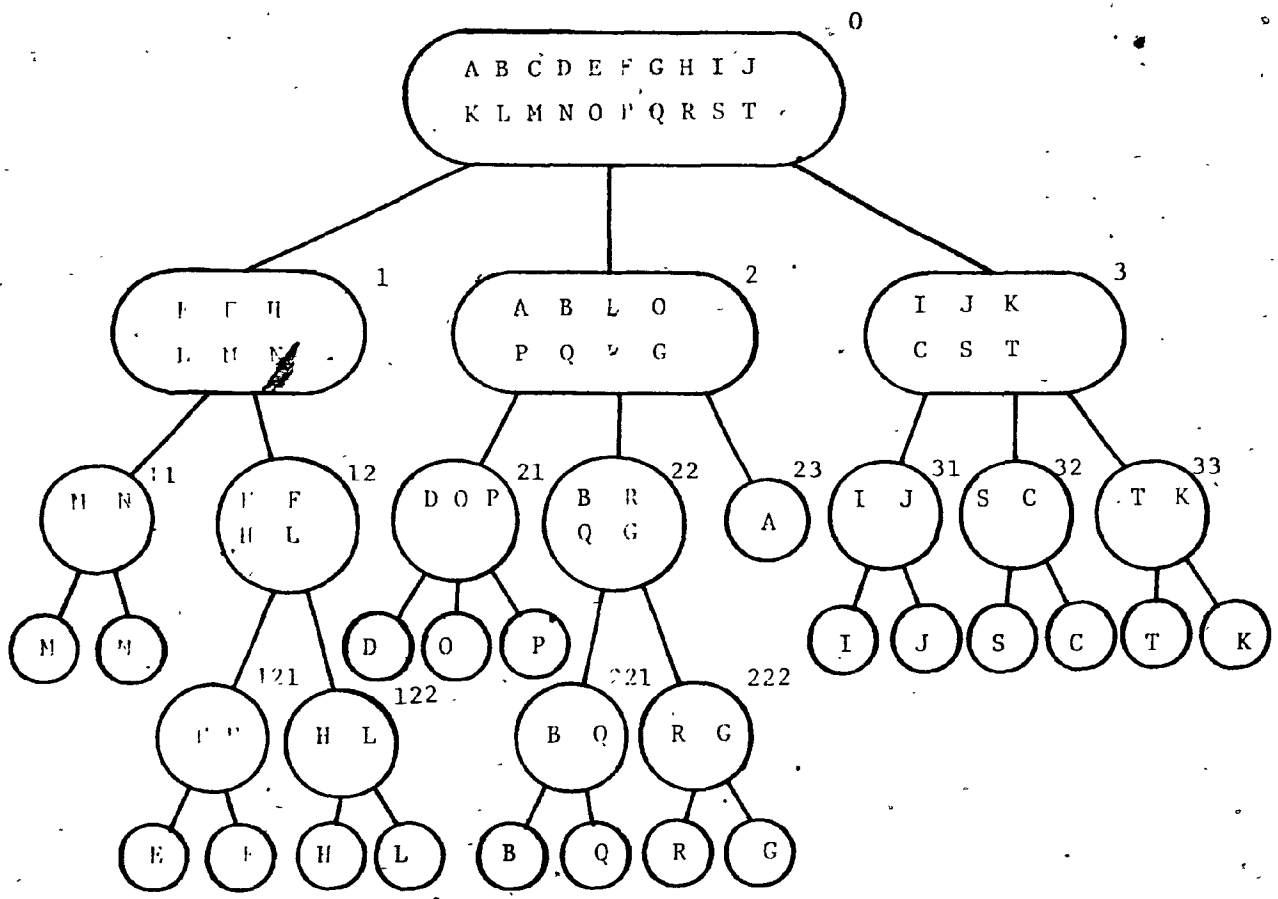



Fig. 2.2 Symbols in a general tree classifier

root is

$$P^{(k-1)} = \sum_{i=1}^{m_k} p_i^{(k)} \tag{2.23}$$

According to Shannon's entropy measure,

$$H_n(P_1, P_2, \ldots, P_n) = -\sum_{i=1}^{n} P_i \log P_i \tag{2.24}$$

where $P_i$'s (i=1,2,...,n) are probabilities,

$$\sum_{i=1}^{n} P_i = 1, \quad 0<P_i<1, \quad i=1, 2, \ldots, n. \tag{2.25}$$

Although there are some more general entropy measures with Shannon's as their special case, such as Renyi's entropy of order $\alpha$, generalized entropy of degree $\alpha$ and R-norm entropy [2.10, 2.11], Shannon's is preferred because it is the only one which satisfies the strong additivity in the following form [2.9,2.10],

$$H_l(p_1q_{11}, \ldots, p_1q_{1n_1}, p_2q_{21}, \ldots, p_2q_{2n_2}, \ldots, p_mq_{mn_m})$$
$$= H_m(p_1, p_2, \ldots p_m) + \sum_{i=1}^{m} p_i H_{n_i}(q_{i1}, q_{i2}, \ldots, q_{in_i}) \tag{2.26}$$

where $l=\sum_{i=1}^{m} n_i$, and the probabilities p's and q's satisfy

$$\sum_{i=1}^{m} p_i = 1$$

$$\sum_{j=1}^{n_i} q_{ij} = 1, \quad i=1,2,\ldots,m.$$

As is known, the searching time in a complete binary decision tree is log n units, where n is the number of terminal nodes and one unit time is that for an unknown pattern to pass one level of the tree [1.7]. For a general tree, the result is very similar to (2.17) and (2.18) given by Shannon. For relating the pattern recognition problem with entropy reduction, we give the result in the following form.

THEOREM 2.1. Suppose $H_0$ is a positive constant. If there is no overlap in the tree and the quantity

$$H^{(k)} = -\sum_{i=1}^{m_k} p_i^{(k)} \log p_i^{(k)} \geq H_0 \qquad (2.27)$$

for each internal node at any level k, then the average time for the unknown pattern to reach the terminals from the root satisfies

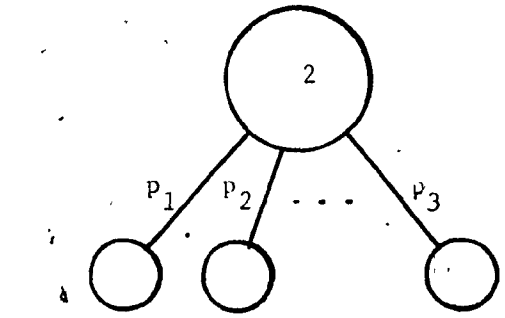
$$\overline{T} \leq H/H_0 \text{ unit} \qquad (2.28)$$

where $H = H_n(P_1, P_2, \ldots, P_n)$.

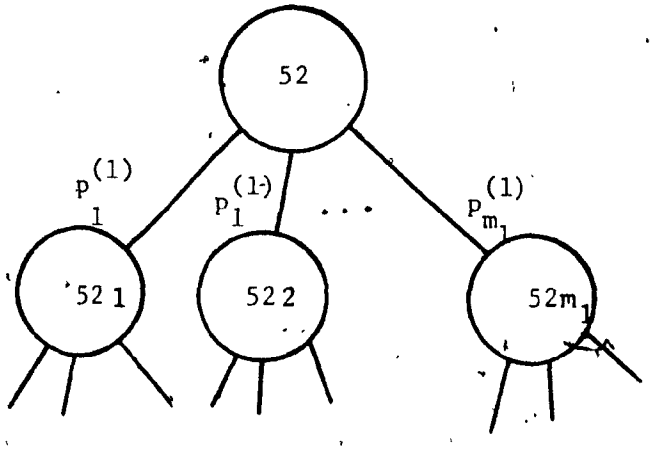
PROOF.

Note that each internal node with all its descendants is actually a (sub-) decision tree. Let us prove (2.28) by induction based on the number of levels of the decision tree.

For a one level decision tree such as the one shown in Fig.2.3a.




a) A One Level Tree




b) An k+1 Level Tree

FIG. 2.3 Induction in the Tree

$$P_i^{(1)} = P_i,\ i=1,2,\ldots,n.$$

Since $-\sum_{i=1}^{n} P_i^{(1)} \log P_i^{(1)} \geq H_0$

we have directly

$$H = -\sum_{i=1}^{n} P_i \log P_i \geq H_0$$

While the searching time is one unit, which satisfies

$$\bar{T} = 1 \leq H/H_0.$$

Now suppose eqn. (2.28) holds for any tree or subtree with the number of level less than or equal to K. For a tree with (K+1) levels as shown in Fig.2.3b, suppose the entropy of the i-th subtree is $h_i$, we have

$$\bar{T} = 1 + \sum_{i=1}^{m_1} P_i^{(1)} \bar{T}_i$$

$$\leq (H_0 + \sum_{i} P_i^{(1)} H_i)/H_0$$

$$\leq [-\sum_{i=1}^{m_1} P_i^{(1)} \log P_i^{(1)} -$$

$$\sum_{i=1}^{m_1} P_i^{(1)} \left(\sum_{j\in\omega_i} (P_j/P_i^{(1)}) \log (P_j/P_i^{(1)})\right)]/H_0$$

$$= -\sum_{i=1}^{m_1} \sum_{j\in\Omega_i} P_j \log P_j /H_0$$

$$= -\sum_{j=1}^{n} P_j \log P_j /H_0$$

$$= H/H_0 \qquad \text{QED.}$$

This result means that, the time requirement of a general decision tree is in the order O(H), or O(log n) when all $P_i$'s have the same values.

THEOREM 2.2. Under the assumption of Theorem 2.1, if the error rate produced at each node of any level k (i.e., the error probability for the unknown pattern to go to a wrong node) is

$$e^{(k)} \leq e_0$$

for some constant $e_0$, then the error rate of this decision tree is

$$e \leq e_0 \cdot H/H_0 \qquad (2.29)$$

Proof.

For those unknown patterns going through $K_i$ levels before reaching the terminal node, the error rate is

$$e_i \leq e_0 K_i.$$

the eror rate of the tree is then the expectation

$$e = E(e_i) \leq e_0 E(K_i)$$

By Theorem 2.1, (2.29) is obtained. Q.E.D.

The intuition of this result is as follows. According to the condition of this theorem, the entropy reduced at each level is not less than $H_0$, and the error rate produced at each level is not greater than $e_0$. Since the total entropy to be reduced is H, the average number of levels for the unknown pattern to pass through is $H/H_0$, the overall error rate can be expected to lie within the bound given in eqn. (2.29). The reason that this bound is much larger than $e_0$ is that, whenever an unknown pattern goes to a wrong branch from any internal node along its path, it will never come back to the correct terminal. We call this phenomenon the error accumulation effect. This is obviously a serious problem, especially when H is large, as in the case of large character set recognition. There are two ways to reduce the overall error rate. The first is to minimize $e_0$ by allowing overlap, but this will cause some other problem when the number of classes is very large, as will be analyzed in the next section. The second is to improve the tree search so as to eliminate the "error accumulation effect", which is proposed in this thesis and described in section 2.4.

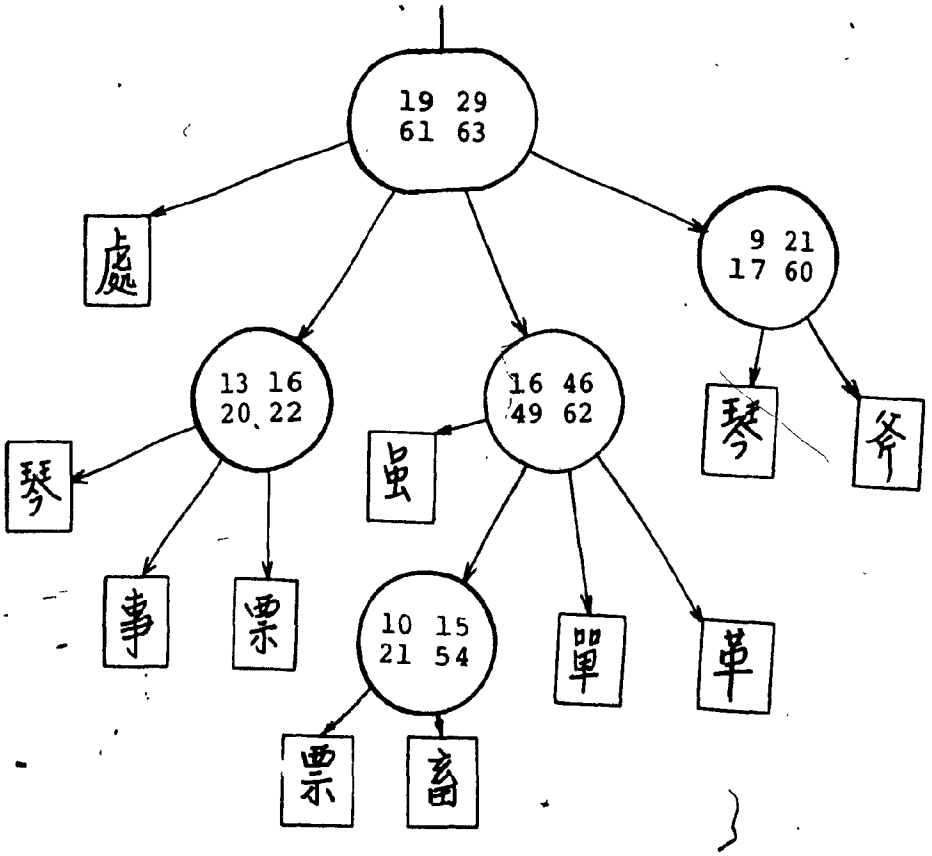
## 2.3. Decision Tree with Overlap

Overlap was discussed in refs.[1.9,1.14] and some other papers. We can use Fig.2.4a to explain it, where the decision tree classifies a group of Chinese characters. One can see that both characters " 琴 " and " 栗 " appear in more than one terminal node, which makes the number of terminals larger than the number of classes. We say that this decision tree has 2 overlaps. On further examination of this tree as shown in Fig.2.4b, one can find the reason for these overlaps, i.e. both " 琴 " and " 栗 " go to more than one child node from the first level to the second level of the tree producing 2 overlaps at the root node.

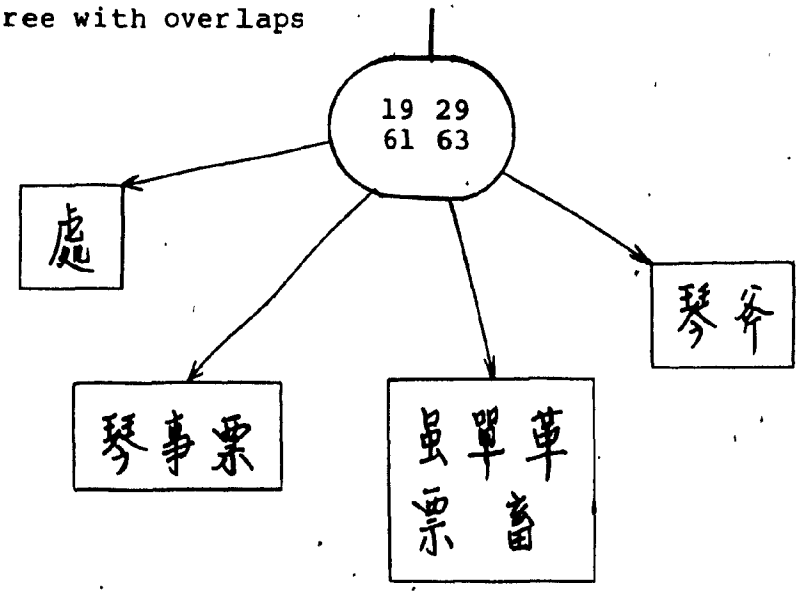
For many practical reasons, overlap happens very often in a decision tree. When the number of classes is very large, overlap may create a very serious problem. For example, in ref.[1.14], the tree classifying 3155 chinese Characters had about 15000 overlaps. Obviously overlap degrades efficiency in terms of time and storage of the tree and should therefore be suppressed in large character set problems.

In the analysis in this section the following definitions are used:

DEFINITION 2.1. Suppose in a decision tree with overlap, the nodes at the second level contain $n_1, n_2, \ldots, n_{m_1}$




a. a tree with overlaps




b. overlaps occurring at level 2

Fig. 2.4 Overlaps in a Decision Tree.

classes respectively, where $\sum_{i=1}^{m_1} n_i > n$. The reduction of entropy at this level is defined as

$$\Delta H = -\sum_{j=1}^{n} P_j \log p_j + \frac{1}{n} \sum_{i=1}^{m_1} n_i \sum_{j\in\Omega_i} (P_j/P_i^{(1)}) \log (P_j/P_i^{(1)}) \qquad (2.30)$$

where $P_j$, $P_i^{(1)}$ and $\Omega_i$ have the same meaning as in eqns. (2.20,2.21) except may be

$$\Omega_i \cap \Omega_j \neq \Phi \text{ for some } i \neq j.$$

One can verify that when $P_1=P_2=....=P_n=1/n$,

$$\Delta H = \log n - (\sum_{i=1}^{n} n_i \log n_i)/n. \qquad (2.31)$$

DEFINITION 2.2. The value OP is defined as the number of overlaps at the node plus 1:

$$OP = \text{number of overlaps} + 1 \qquad (2.32)$$

We add 1 to the right hand side of eqn.(2.32) to account for the extra memory used for the parent node. For example, in Fig.2.4b, there are 2 overlaps, and OP = 3. For OP we have the following theorem.

THEOREM 2.3. If the overlaps happening at each node of level 1 are

$$OP^{(1)} \leq n^{(1)} \Delta H^{(1)} / G_0 \qquad (2.33)$$

where $n^{(1)}$ is the number of classes in the sub-decision tree rooted at the node, $\Delta H^{(1)}$ as defined in eqn.(2.31)

for this node, and $G_0$ is some constant (minimum Gain), then the total number of overlaps plus the number of internal nodes of the tree is

$$OP \leq n.H / G_0 \qquad (2.34)$$

PROOF. For a one level tree, eqn.(2.34) is trivial since OP=1 in both eqns.(2.33) and (2.34). Suppose (2.34) holds for any tree or subtree with the number of levels less than or equal to K, then for a (K+1) level tree,

$$OP = OP^{(1)} + \sum_{i=1}^{m_1} OP_i$$

where $OP_i$ is the OP value of the subtree rooted at the ith child node. By eqn.(2.33) and the induction assumption

$$\begin{aligned} OP &< (n.\Delta H^{(1)} + \sum_{i=1}^{m_1} n_i H_i)/G_0 \\ &= [\, n(-\sum_{j=1}^{n} P_j \log P_j + \\ &\quad \frac{1}{n} \sum_{i=1}^{m_1} \sum_{j\in\Omega_i} (P_j/P_i^{(1)}) \log (P_j/P_i^{(1)})) \\ &\quad -\sum_{i=1}^{m_1} n_i \sum_{j\in\Omega_i} (P_j/P_i^{(1)}) \log (P_j/P_i^{(1)})]/G_0 \\ &= n.H / G_0 \end{aligned}$$

which completes the proof. QED.

When $P_1=P_2=\ldots=P_n=1/n$, we have

$H = \log n$, and

$$OP \leq n \log n / G_0 \qquad (2.35)$$

If eqn.(2.33) holds at each node of level 1, eqn.(2.35) shows that OP can be expected to be within the order O(n log n). Furthermore one can see it is important to minimize the value of Gain

$$G = (n^{(1)} \Delta H^{(1)}) / OP^{(1)} \qquad (2.36)$$

at each node of level 1 in the design of the decision tree. If we modify definition 3.1 as follows, we can get a tighter estimation of OP.

DEFINITION 2.3. Using the notations

$$H = -\sum_{j=1}^{n} P_j \log P_j$$

and

$$H_i = -\sum_{j \in \Omega_i} (P_j / P_i^{(1)}) \log (P_j / P_i^{(1)}),$$

the entropy reduction is defined as

$$\Delta \tilde{H} = H - \sum_{i=1}^{m_1} \exp(H_i - H) H_i \qquad (2.37)$$

Similar to definition 2.2, when

$$P_1 = P_2 = \ldots = P_n,$$

we have

$$\Delta \tilde{H} = \log n - (\sum_i n_i \log n_i) / n.$$

THEOREM 2.4. If

$$OP^{(l)} \leq \Delta\tilde{H}^{(l)} \exp(H^{(l)})/G_0 \qquad (2.38)$$

at each node of level l, then

$$OP \leq H \exp(H) / G_0 \qquad (2.39)$$

Proof. Similar to Theorem 2.3, we have

$$\begin{aligned} OP &\leq [\exp(H)\, \Delta\tilde{H}^{(l)} + \sum_{i=1}^{m_l} \exp(H_i) H_i] / G_0 \\ &= \exp(H) [H - \sum_i \exp(H_i - H) H_i \\ &\quad + \Sigma \exp(H_i - H) H_i]/G_0 \\ &= H \exp(H) / G_0. \end{aligned}$$

QED.

In general

$$H \leq \log n$$

and

$$\exp(H) \leq n,$$

which means (2.39) is a tighter form than (2.34). But by eqn.(2.38), the quantity to be controlled at each node is

$$G = \Delta\tilde{H}^{(l)} \exp(H^{(l)})/OP^{(l)} \qquad (2.40)$$

which is more complex than that given in eqn.(2.36).

For the special case

$$P_1 = P_2 = \ldots = P_n = 1/n,$$

we arrive at the same result as in (2.35).

Now we are ready to illustrate some results about searching time and error rate for a decision tree with overlap.

THEOREM 2.5. If at each node of any level k of the decision tree,

$$\tilde{H} = H^{(k)} - \sum_{i=1}^{m_1} P_i^{(k)} H_i^{(k)} \geq H_0 \qquad (2.41)$$

where $H^{(k)}$ and $H_i^{(k)}$ have the same meaning as H and $H_i$ in definition 2.3 and $H_0$ is a constant, then the average searching time is

$$\overline{T} \leq H / H_0 \qquad (2.42)$$

PROOF. Following the proof in Theorem 2.1, we have

$$\begin{aligned}
\overline{T} &= 1 + \sum_{i=1}^{m_1} P_i^{*(1)} \overline{T}_i \\
&\leq 1 + \sum_{i=1}^{m_1} P_i^{(1)} \overline{T}_i \\
&\leq (H_0 + \sum_{i=1}^{m_1} P_i^{(1)} H_i) / H_0 \\
&< [(H - \sum_i P_i^{(1)} H_i) + \sum_i P_i^{(1)} H_i] / H_0 \\
&= H / H_0,
\end{aligned}$$

where $P_i^{*(1)}$ is the probability for an unknown to go to the ith branch, which is not greater than $P_i^{(1)}$.

QED.

Similar to Theorem 2.2, it is easy to prove the following :

THEOREM 2.6. Under the assumption of Theorem 2.5., if the error rate at each node of level k satisfies

$$e^{(k)} \leq e_0, \tag{2.43}$$

then the error rate of the decision tree is given by

$$e \leq e_0 . H / H_0. \tag{2.44}$$

One can see from the proofs of the above theorems that, eqns. (2.28), (2.29), (2.34), (2.39), (2.42) and (2.44) give tight estimation, i.e., each equality can be approached in special cases under the conditions in the corresponding theorems. In fact these results show us that, the overlap is in the order O(n log n), the error rate and search time are both in the order O(log n). They help us to set some principles in designing the decision trees.

## 2.4. Design Principles

Assume we have to design a decision tree for an n-class problem, each class having a-priori probability $P_i$, i=1,2,...,n. Also some features have been extracted from a given set of training samples. The average number of child nodes of each internal node is called branch factor, B. For example, for a complete binary tree, B=2.

A) Optimal Value of B.

In many applications, a binary tree is assumed. But B=2 may not be the optimal choice. Suppose

$$P_1 = P_2 = .... = P_n$$

then the value of $H_0$ in eqn.(2.42) is

$$H_0 = \log B$$

and the equality is reached

$$\overline{T} = B \log n / \log B.$$

It is easy to show that $\overline{T}$ is minimized when B=3. When B=2 and 4, $\overline{T}$ has the same value. But less internal nodes are used when B=4 than B=2. Based on these considerations

$$3 \leq B \leq 4 \qquad (2.45)$$

is preferred, and a binary tree is not the optimal type of

tree in the case.

Under the above assumption, eqn.(2.44) becomes

$$e \leq e_0 \log n / \log B \qquad (2.46)$$

which implies that a bigger value of B is preferred. A combination of eqns.(2.45) and (2.46) indicates that

$$3 \leq B \leq 5 \qquad (2.47)$$

may be the proper range of B. In the design of the decision tree, the branch number at each internal node does not necessarily have the same value and eqn. (2.47) provides flexibility.

B) Clustering Objective

In the design of the decision tree, a clustering algorithm is needed in a broad sense. Let us discuss what should be the objectives of clustering.

The memory requirement of the decision tree is

$$M = n + OP \qquad (2.48)$$

by Definition 2.2. According to Theorem 2.3 or 2.4, this could be in the order

$$M \simeq n + n \log n / G_0 \qquad (2.49)$$

If n is very large, n log n will be much larger than n. In order to control the memory requirement of the tree, it

is important to increase the value of $G_0$. This is related to feature extraction and selection, which will be discussed in Chapter 4. The problem in clustering is how to control $G_0$ for a given group of features. As can be seen from eqn.(2.38), if the ratio of entropy reduction over the quantity OP is maximized at each internal node and each level, the value of $G_0$ is then maximized. Based on this consideration, the overall objective of clustering should be

$$\text{MAX Gain} = n^{(l)} \Delta H^{(l)} / OP^{(l)} \qquad (2.50)$$

at each internal node in the design phase, so that the value of $G_0$ is maximized and the memory requirement is suppressed. Meanwhile the depth of the tree, hence also the average searching time and error rate, are reduced.

c) Error Rate Control.

When an unknown pattern passes through each level of the tree, the error rate occurring at each level is accumulated. The error rate of the entire tree is in the order O(H) or O(n log n) by eqn.(2.44). This is the main disadvantage of the tree classifier and it becomes very serious when n is very large. The reason that error accumulates can be explained as follows. In the search of the tree, whenever an error occurs at a certain level, the unknown pattern will go to a wrong branch and never come back to the right one. This difficulty can be solved by recording some "historical

information" of the search. Later on, when it is found that the current branch or terminal node is not likely to be the correct one, this "historical" information can be used for backtracking along the tree. In this way, the error accumulation problem may be solved, which will be discussed in Chapter 5.

## 2.5. Conclusion

In addition to Bayes' decision theory, the entropy reduction approach has shown to be useful in the analysis of a tree classifier. The theorems proved in this chapter show that its searching time and error rate are both in the order O(H) and the memory requirement in the order $O(H \cdot \exp(H))$. Although the time efficiency of the tree classifier is satisfactory, there are serious problems with both error rate and memory requirement. To solve these problems, some design principles have been proposed. Chapters 3, 4 and 5 show the application of these principles to solve these problems. In chapter 6, some simulation experiments will be presented with encouraging results, which justify the theoretical results as well as the proposed design principles described in this chapter.

## 2.6. Appendix

First, we calculate

$$I = \int_{\Omega_i} p(\underline{X}|\ \omega_i)\ dx$$

$$= \int_{\Omega_i} (2\pi)^{-m/2}\ (|\Sigma_i|)^{-1/2}$$

$$\exp[-(\underline{X}-\underline{\mu}_i)^t\ \Sigma_i^{-1}\ (\underline{X}-\underline{\mu}_i)/2]\ d\underline{X}$$

for $i=1,2,\ldots,n$.

We can find an orthonormal transformation $T_i$,

$$\underline{X} = T_i\ \underline{Y} + \underline{\mu}_i \qquad (2.51)$$

such that the covariance matrix $\Sigma_i$ is diagonalized in the new system

$$\tilde{\Sigma}_i = \text{diag}\ (\sigma_1,\sigma_2,\ldots,\sigma_m). \qquad (2.52)$$

In this way the integral I becomes

$$I = \int_{\omega_i} (2\pi)^{-m/2}(\sigma_1.\sigma_2\ldots\ldots\ \sigma_m)^{-1}$$

$$\exp[-(y_1^2/\sigma_1^2+\ \ldots+y_m^2/\sigma_m^2)/2]\ d\underline{Y} \qquad (2.53)$$

where

$$\omega_i = \{\underline{Y}|\ y_1^2/\sigma_1^2+\ldots.\ \ Y_m^2/\sigma_m^2 < \alpha\ \}. \qquad (2.54)$$

Let

$$y_1 = r\ \sigma_1\ \cos\theta_1$$

$$y_2 = r\,\sigma_2 \sin\theta_1 \cos\theta_2$$

$$y_3 = r\,\sigma_3 \sin\theta_1 \sin\theta_2 \cos\theta_3$$

$$\int$$

$$\cdots\cdots\cdots\cdots\cdots$$

$$y_{m-1} = r\,\sigma_{m-1} \sin\theta_1 \sin\theta_2 \cdots \operatorname{Sin}\theta_{m-2} \cos\theta_{m-1}$$

$$y_m = r\,\sigma_m \sin\theta_1 \sin\theta_2 \cdots \operatorname{Sin}\theta_{m-1}, \qquad (2.55)$$

then

$$\Omega_i = \{(r, \theta_1, \theta_2, \ldots, \theta_{m-1}) \mid$$

$$0 \leq r < \alpha,\ 0 \leq \theta_1 < \P,\ \ldots,\ 0 \leq \theta_{m-2} < \P,$$

$$0 < \theta_{m-1} < \P\ \}. \qquad (2.56)$$

The Jacobean of (2.55) is

$$J = r^{m-1} \sin^{m-2}\theta_1 \cdots \operatorname{Sin}\theta_{m-2}\,\sigma_1 \cdots \sigma_m \qquad (2.57)$$

and (2.53) becomes

$$I = C_m \int_0^\alpha r^{m-1} \exp(-r^2/2)\,dr \qquad (2.58)$$

where

$$C_m = \begin{cases} (2/\P)^{1/2}/(m-2)!! & \text{when } m \text{ is odd} \\ 1/(m-2)!! & \text{when } m \text{ is even} \end{cases} \qquad (2.59)$$

A further calculation gives

$$I = 1-\exp(-\alpha^2/2)[\alpha^{m-2}/(m-2)!!+\ldots+\alpha^2/2!!+1.]$$

m even,

$$2\Phi(\alpha) - (2/\pi)^{1/2} \exp(-\alpha^2/2) [\alpha^{m-2}/(m-2)!! + \ldots + \alpha^3/3!! + \alpha] \qquad m \text{ odd} \qquad (2.60)$$

where

$$\Phi(\alpha) = (\pi)^{-1/2} \int_0^{\alpha} \exp(-t^2/2)\, dt. \qquad (2.61)$$

## 2.7. References


[1]. K. Fukunaga, Introduction to Statistical Pattern Recognition, Academic Press, New York, 1972.

[2]. R. O. Duda and P. E. Hart, Pattern Classification and Scene Analysis, John Wiley & Sons Inc., New York, 1973.

[3]. T. M. Cover and T. J. Wagner, "Topics in statistical pattern recognition," pp.15-46, in Digital Pattern Recognition, K. S. Fu (ed.), Springer-Verlag, Berlin, Heidelberg, 1976.

[4]. P. A. Devijver and J. Kittler, Pattern Recognition, A Statistical Approach, Prentice-Hall International, Inc., London, 1982.

[5]. K. S. Fu, Syntactic Pattern Recognition and Applications, Prentice-Hall Inc., Englewood Cliffs, N.J. 1982.

[6]. S. Watanabe, "Panel discussion on entropy and pattern recognition," Proc. 5th Int. Conf. on Pattern Recognition, p.1276, Dec. 1980.

[7]. S. Watanabe, "Pattern recognition as a quest for minimum entropy," Pattern Recognition, vol. 13, No.5, 381-387, 1981.

[8]. C. E. Shannon, "A Mathematical Theory of Communication," Bell Sys. Tech. J. vol. 27, 379-423,

623-656, 1948.

[9]. S. Kullback, Information Theory and Statistics, Dover Publication, Inc., New York, 1968.

[10]. J. Aczel and J. Daroczy, On Measure of Information and their Characterizations, Academic Press, New York, 1975.

[11]. D. E. Boekee and J. C. A. Van Der Lubbe, "The R-norm information measure," Information and Control, vol. 45, 136-155, 1980.

[12]. P. A. Devijver, "Decision theoretic and related approaches to pattern classification," pp.1-34, in Pattern Recognition, Theory and Applications, K. S. Fu (ed.), Noordhoff-Leyden, The Netherlands, 1977.

[13]. C. H. Chen, "A review of statistical pattern recognition," in Pattern Recognition and Signal Processing, C. H. Chen (ed.) pp.117-132., Sijthoff Noordhoff, Alphen aan den Rijn-The Netherlands, 1978.

[14]. L. Kanal, "Patterns in pattern recognition : 1968-1974," IEEE Trans. On Information Theory, vol. IT-20, No.6, pp.697-722, Nov. 1974.

[15]. Q. R. Wang and C. Y. Suen, "Analysis of decision tree and its application to large character set recognition," in press, IEEE Trans. PAMI., vol. 6. 1984.

# CHAPTER THREE

## ISOETRP — A CLUSTERING ALGORITHM WITH NEW OBJECTIVES

### Introduction

A new clustering algorithm ISOETRP has been developed. Several new objectives have been introduced to make ISOETRP particularly suitable to hierarchical pattern classification. These objectives are : a) minimizing overlap between pattern class groups, b) maximizing entropy reduction, and c) keeping balance between these groups. The overall objective to be optimized is

Gain = Entropy Reduction / (Overlap +1).

Balance is controlled by maximizing the Gain. An interactive version of ISOETRP has also heen developed by means of an overlap table. It has been shown that ISOETRP gives much better results than other existing algorithms in optimizing the above objectives.

## 3.1. Clustering in Tree Classifier Design

In each internal node of the tree classifier there are certain classes which can be divided into smaller groups, each of which can be assigned to a child node. In the design of the tree, this process is repeated again and again until only one class remains at each terminal node. A clustering algorithm is necessary in the whole process. Since it is applied at the internal nodes thousands of times in the design of a large tree, this algorithm is very important.

Different from traditional clustering algorithms (like those presented in refs. [3.4, 3.7, 3.10]), each sample here is a pattern class instead of a pattern. Broadly speaking, a pattern class is a random variable or vector, which lies in the d-dimensional feature space with certain distribution density, as stated in Chapter 2. In parametric approach, this region of the class can be $\Omega_i$ in eqn.(2.10), where $\alpha$ is chosen big enough, e.g. larger than 3.50. In non-parametric approach, this can be considered as some region in the d-dimensional space, which properly covers all training samples belonging to this class. In clustering, each pattern class may be assigned to 2 or more neighboring clusters. Since each class occupies a finite volume in space, which is different from a sample point, overlap may

occur, i.e. two or more clusters may have some classes as their common members. This is illustrated in Fig.3.1, where letters G and R constitute the overlap between two clusters. Overlap was considered as an open problem in [3.10]. This chapter is devoted to a new clustering algorithm, which deals with overlap problem and gives a practical solution. In section 3, it will be seen that "overlap" can even help the human designer to implement this algorithm.

The concept of entropy reduction in tree classifier has been proved powerful in the analysis presented in Chapter 2. It will play an equally important role in this clustering algorithm, especially in the case when the number of classes is very large and there is no easy way to calculate Bayes' recognition rate at each individual level.

Tree balance is also important, which has been reported in many papers about decision tree, e.g. [1.9]. If the tree is not well balanced, then some subtrees may become much bigger than the others, which makes the tree depth larger than that of a balanced one. But the balance factor is reflected in entropy reduction quantity, which will be explained in section 2. This makes it possible to take a consistent treatment in the tree classifier design.

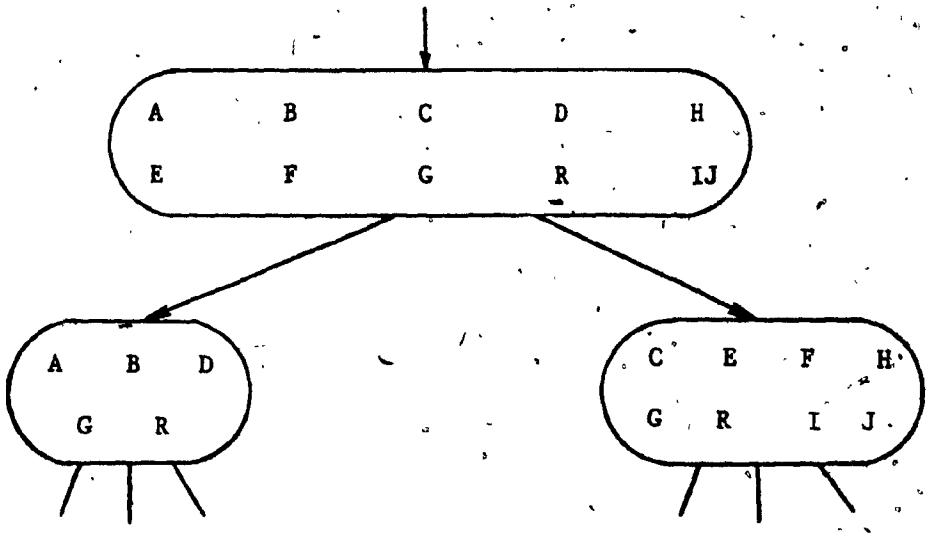



FIG. 1. Overlaps (G,R) in the tree classifier

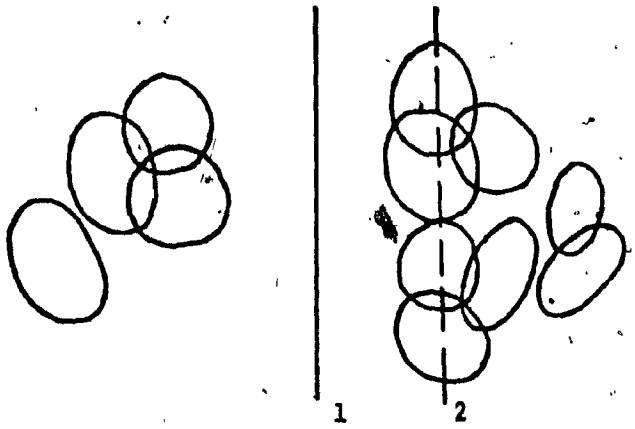



FIG. 2. OVLP and BLNC

## 3.2. Cluster Objectives

To start with, let us consider two well-known clustering algorithms, ISODATA and KMEANS, which aim at minimizing the "performance index" [3.6] defined as follows,

$$PIDX = \sum_{j=1}^{k} \sum_{i=1}^{n_j} d^2_{jj_i} \qquad (3.1)$$

where k is the number of clusters, $n_j$ the number of samples within the jth cluster, $d_{jj_i}$ the distance between the jth cluster center and the ith sample in the jth cluster. PIDX is totally different from either the entropy or overlap, which implies that the above 2 clustering algorithms do not serve the tree design very well.

The objectives of ISOETRP proposed in [3.8] are described in this section.

A) ENTRP

Suppose there are n classes $\omega_1, \omega_2, \ldots \omega_n$ in the original samples,

$$\Omega = \{ \omega_i \mid i = 1, 2, \ldots, n \} \qquad (3.2)$$

each associated with a-priori probability $P_i$, then the original entropy can be expressed as

$$E_1 = - \sum_{i=1}^{n} P_i \log P_i \qquad (3.3)$$

After clustering, they are divided into k smaller groups,

each containing some classes. These smaller groups are denoted by

$$G_j = \{ \omega_{j_l} \mid l = 1, 2, \ldots, n_j \}$$

$$j = 1, 2, \ldots, k \qquad (3.4)$$

with the approximate a-priori probabilities

$$\tilde{P}_j = \sum_{l=1}^{n_j} P_{j_l},$$

$$j = 1, 2, \ldots, k. \qquad (3.5)$$

Note that it is possible that

$$G_{j_1} \cap G_{j_2} \neq \Phi \text{ for some } j_1 \neq j_2$$

and

$$P = \sum_{j=1}^{k} \tilde{P}_j > 1$$

due to overlap.

In each new group, the entropy is

$$E_{2_j} = -\sum_{l=1}^{n_j} (P_{j_l}/\tilde{P}_j) \log (P_{j_l}/\tilde{P}_j),$$

$$j = 1, 2, \ldots, k \qquad (3.6)$$

Since the probability for an unknown to be the jth group is $\tilde{P}_j/P$, the average entropy can be expected as

$$E_2 = \left(\sum_{j=1}^{k} \tilde{P}_j E_{2_j}\right) / P$$

$$= -[\sum_{j=1}^{k} \sum_{l=1}^{n_j} P_{j_l} \log (P_{j_l}/\tilde{P}_j)]/P \qquad (3.7)$$

Entropy reduction is defined as

$$\text{ENTRP} = E_1 - E_2$$

$$= [\sum_{j=1}^{k} \sum_{l=1}^{n_j} P_{j_l} \log (P_{j_l}/\tilde{P}_j)]/P$$

$$- \sum_{i=1}^{n} P_i \log P_i \qquad (3.8)$$

In the special case

$$P_i = 1/n, \quad i = 1, 2, \ldots, n,$$

we have

$$\tilde{P}_j = n_j / n, \; j = 1, 2, \ldots, k$$

$$P = (\sum_{j=1}^{k} n_j) / n$$

and

$$E_1 = \log n \qquad (3.9)$$

$$E_2 = (\sum_{j=1}^{k} n_j \log n_j) / (\sum_{j=1}^{k} n_j) \qquad (3.10)$$

$$\text{ENTRP} = \log n - (\sum_{j=1}^{k} n_j \log n_j) / (\sum_{j=1}^{k} n_j). \qquad (3.11)$$

One can see from the simple formula that, the smaller the new groups are, the larger ENTRP is, meaning that more uncertainty is being removed.

B) OVLP

OVLP is defined as the difference

$$OVLP = \sum_{j=1}^{k} n_j - n \qquad (3.12)$$

which means the number of extra nodes created at the current clustering. The smaller OVLP is, the better the clustering result is. The best case occurs when

$$OVLP = 0$$

and

$$\Sigma n_j = n.$$

C) BLNC

Balance factor can be defined as the deviation of the number of classes in the new groups from the average number of classes

$$BLNC = \left( k.\sum_{j=1}^{k} (n_j - n/k)^2 \right)^{1/2}/n \qquad (3.13)$$

the deviation between $n_j$'s is normalized by their average value n/k. The smaller BLNC is, the more balanced the clustering result is. The best case occurs when BLNC = 0, i.e. all the new groups have the same number of classes.

In ISOETRP, BLNC is not controlled directly, the reason of which is as follows.

Often BLNC conflicts with OVLP. Take the example given in Fig.2. One can see the best result is indicated by line 1, the OVLP of which is zero while

$$BLNC = [2((4-6)^2+(8-6)^2)]^{1/2}/12 = 0.333, \quad (3.14)$$

which is large. If line 2 is used, BLNC becomes zero but

$$OVLP = 4$$

which is much worse than before, as seen in Fig. 3.2. Usually OVLP is optimized or sub-optimized when the clustering result coincides with the intrinsic structure of the real data. In other words, OVLP is more important than BLNC and should be controlled more carefully.

Even if BLNC does not conflict with OVLP, minimizing BLNC is not necessarily the optimal strategy. This is illustrated in the following 2 cases. In the first case where all $P_i$'s in eqn.(3.3) are not equal, a balanced tree is usually not the optimal one. Similar to Shannon's coding theorem [2.8], the optimal one minimizes ENTRP in eqn.(3.8). In the second case where the $P_i$'s are equal, ENTRP = $E_1 - E_2$ is used. As is well known, for the given number of groups, k, the more balanced the groups(or $n_i$'s) are, the larger the value of ENTRP is. If $n_i$'s are not balanced at all, say some $n_i$ is too large, $n_i \simeq n$, then we have ENTRP $\simeq$ 0 by (3.10), which is also far from being maximized. In both cases, ENTRP should be maximized, leading to the

minimization of BLNC in the second case.

d) GAIN

The above analysis indicates that we have to control ENTRP and OVLP. The overall objective of ISOETRP is measured by

$$GAIN = ENTRP / f(OVLP) \tag{3.15}$$

where f is a monotonic increasing real function.

As has been proved in Theorem 2.3 and eqn.(2.36), if we take

$$f(OVLP) = OVLP + 1, \tag{3.16}$$

the entropy reduction per unit memory can be optimized by GAIN in (3.15). In this way, the performance of the tree classifier in terms of memory requirement, searching time and error rate can be optimized, this has been proved in Chapter 2.

In summary, ISOETRP aims at maximizing the GAIN, i.e., the ratio of ENTRP over (OVLP + 1), where BLNC is considered by controlling the entropy reduction ENTRP.

## 3.3. Data Structure and Procedures

A) Data Structure

The data describing the pattern classes are stored in matrices. Information on class center, shape and size of the region occupied by each class and so on, is included.

The current clustering result is stored in a record CRNT of type RSLT, the definition of which is as follows when written in Pascal:

```
TYPE
    CLTS = packed array[1..MCLT, 1..MAX] of boolean;
    CTRS = array[1..MCLT, 1..NOF] of real;
    OLPS = array[0..MCLT, 0..MCLT] of integer;
    PRSN = array[1..MCLT] of boolean;
    RSLT = record
             NCLT : integer;
             CLT : CLTS;
             CTR : CTRS;
             OLP = OLPS;
             DEDUCE, BALANCE, GAIN : real
             MBS : array[1..MAX] of integer
           end;
```

where the constants are,

MAX — maximum number of classes in the original group,

MCLT — maximum number of clusters,

NOF — number of features used in clustering,

and each array or simple variable has the following meaning :

CLTS : indicating the membership of a class in a cluster,

CTRS : centers of clusters,

OLPS : overlap table, which will be described in the next section,

PRSN : indicating the presence of each cluster,

MBS : indicating to which cluster a class mainly belongs,

DEDUCE, BALANCE and GAIN : as explained in the above section.

In addition to CRNT, the previous 5 best clustering results are stored in an array RLTS according to the value of GAIN. Each entry of this array has the same record type as CRNT.

B) Update Procedures

The procedures in ISOETRP are described as follows, some of which are also illustrated in Fig.3.3.

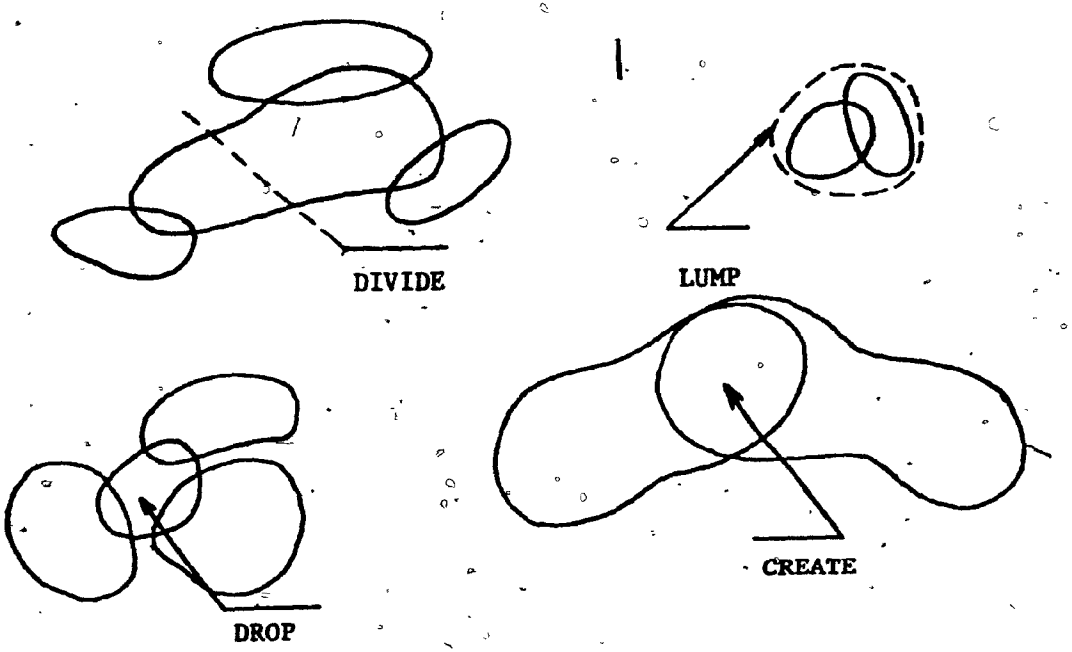



FIG. 3.3 Update Procedures

Procedure KMEANS;

{ calculate the center of each cluster }

procedure DISTRIBUTE;

{ assign each class to some nearest cluster ( or clusters in case of overlap ) and then compute the new centers }

begin

for every class do

1. Calculate the distance between the class center and all other cluster centers.

2. Assign this class as NORMAL to the nearest cluster, the distance of which is denoted by T;

3. Assign it as OVERLAP to any other cluster with distance TT, if TT < T + $\alpha$ * DEV where DEV is the deviation of the class in the direction towards this cluster center, $\alpha$ is a pre-specified value

end;

Procedure PRINCIPAL;

{ find the principal axis of a cluster }

begin

1. Calculate the within cluster covariance matrix A;

2. Calculate the largest eigen value $\lambda_1$ and the corresponding eigen vector of A, $V_1$.

{ This can be done by the power method, see [3.5]. Since the accuracy of these values is not very important, a few iterations in the power method will suffice }

end;

Procedure DIVIDE;

{ divide a cluster if it is big and/or has many overlaps with other clusters }

begin

Call PRINCIPAL to calculate its principal axis $V_1$ and eigen value $\lambda_1$;

Create two new centers { for new clusters }

$C_1 = C_0 + \beta \lambda_1^{1/2} V_1$;

$C_2 = C_0 - \beta \lambda_1^{1/2} V_1$;

where $C_0$ is the old center and dropped after the new centers are created, $\beta$ is a constant between

0.5 and 1.0;

Call DISTRIBUTE followed by KMEANS for m times { usually $5 \leq m \leq 20$ }

end;

Procedure LUMP;

{ lump two clusters together if they are small and/or they contain many common classes (overlap) }

begin

$C = (C_1 * n_1 + C_2 * n_2) / (n_1 + n_2)$

where $C_1$, $C_2$ are old centers, C the new center and $n_1$, $n_2$ numbers of NORMAL classes in the old clusters

end;

Procedure CREATE;

{ create a new cluster from a pair of clusters, if they are big and contain many common classes }

begin

Calculate the center of these common classes, which is the center of the new cluster

end;

```
Procedure DROP;

    { drop a cluster if it is small and/or contains many
    overlaps }

Procedure RETRIEVE;

    { copy some old results in RLTS to CRNT }

Procedure INITIAL;

    { randomly generates cluster centers using time
    seeds, and update CRNT }
```

## 3.4. Overlap Table and the Interactive Algorithm

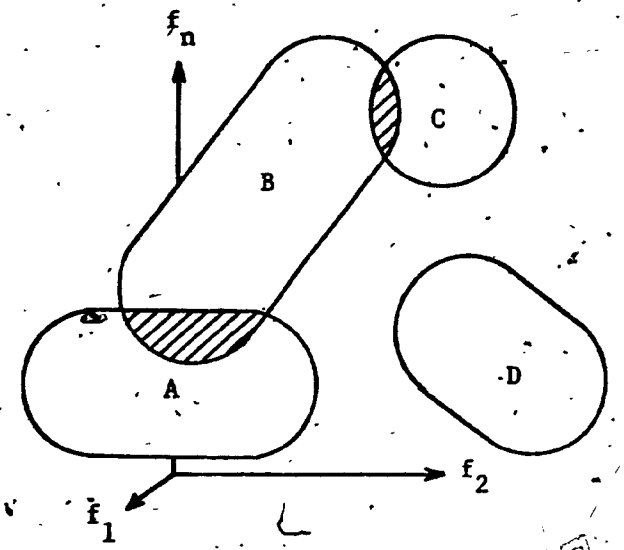
The common functions of the procedure listed in the last section are to increase entropy reduction and decrease overlap. They were included in the first version of ISOETRP [3.8]. Among them, such procedures like KMEANS, LUMP and DISTRIBUTE are similar to those used in the algorithm ISODATA, except that overlap is not considered in the latter. All these procedures were called according to the requirement of reducing entropy and overlap. This program was employed in the work reported in [3.11]. Although it aims at optmizing GAIN, ENTRP and OVLP, often it takes up a lot of time to arrive at a good result. The reason is that no predefined parameters can fit all input data very well, and the operator does not see what is happening to the clusters in a higher dimensional space when the program is running.

A modified version of ISOETRP was developed in [3.9]. The key point is to display the cluster structures and make interaction between the operator and machine possible. Many interactive clustering techniques exist [2.1, 3.2, 3.3], but each of them first maps (metric or non-metric) the multi-dimensional space into a 2-D space and then display the results for further interaction. This degenerative mapping distorts the data space considerably, and the true struture of the clusters can not be visualized easily.

Furthermore, it often takes up a lot of time to find a suitable view of mapping and displaying the 2-D data. In modified ISOETRP, an overlap table is introduced to describe the currnt clustering result in the higher dimensional space. This table enables the operator to decide on an action which will likely bring a better result.

The overlap table is shown in Fig.3.4, where each diagonal element gives the number of classes in the corresponding cluster as a NORMAL member (see Procedure DISTRIBUTE), and the off-diagonal element is the number of overlap classes between the corresponding pairs of clusters. The last row contains an overlap number for each cluster. With the help of this table, one can "see" the cluster structure in the current result and visualize the intrinsic structure of the data. Then he can select one update procedure to instruct the computer to do so.

The flow chart of ISOETRP is shown in Fig.3.5. By INITIAL, the first result is obtained. The overlap table is displayed repeatedly so that the operator can take the appropriate action to improve the result. When the operator finds CRNT is still not satisfactory after some interactions, RETRIEVE and INITIAL can be invoked. Usually MEANS and DISTRIBUTE would have gone through "ITERS" iterations after each of these actions. In this process, whenever a good GAIN is obtained, CRNT is recorded in some entry of RLTS. It terminates after a specified amount of




| | A | B | C | D |
|---|---|---|---|---|
| A | $n_a$ | $n_{ab}$ | 0 | 0 |
| B | $n_{ba}$ | $n_b$ | $n_{bc}$ | 0 |
| C | 0 | $n_{cb}$ | $n_c$ | 0 |
| D | 0 | 0 | 0 | $n_d$ |

FIG. 3.4 The cluster structure and its overlap table

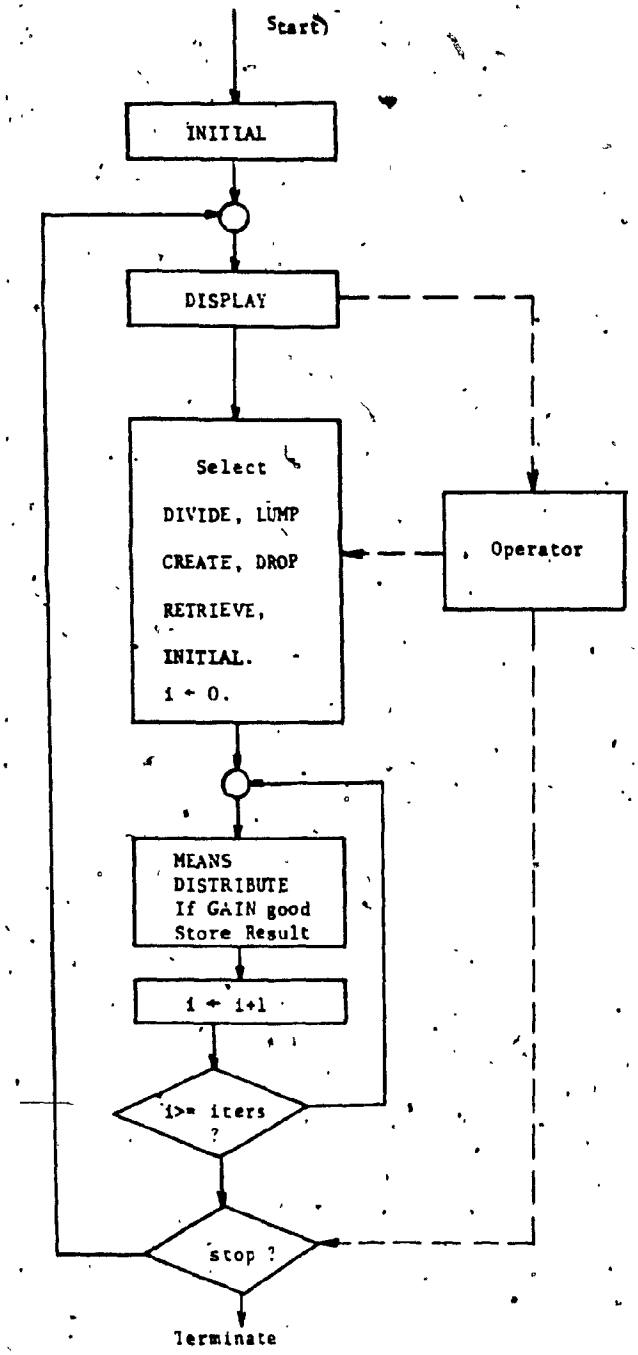



FIG. 35 Flowchart of ISOETRP

effort has been spent without further improvement.

Sample run of ISOETRP is shown in Fig.3.6, where each step is described as follows :

The number of calsses is n = 44. The merit of each feature was calculated by eqn.

$$\text{merit} = \sigma^2 / \left(\sum_{i=1}^{n} p_i \sigma_i^2\right) \qquad (3.17)$$

where $\sigma$ is the overall deviation of the feature, $\sigma_i$ the deviation within the ith class (i=1, 2, ..., n), $p_i$ the a-priori probability of this class. 8 features with the largest merit values were selected. Clustering was made in the corresponding 8-dimensional space. The first result was randomly generated as

OVLP = 10,

ENTRP =1.2208,

GAIN = 0.1110,

BALANCE = 0.5475.

According to the overlap table in each iteration, the operator instructed the computer to do the following step by step,

?G12 --- create a new cluster from the overlapped cluster pair 1 and 2,

```
          FEATURES
          NO.     MERIT
           1     80.54151
           2     92.55690
          33     61.31678
          34     62.48674
          61    107.30967
          63    118.96626
          64     61.92409
          95     75.65685
 ( 1 )( 2 )( 3 )( 4 )( 5 )
  18     5      1      1      2      21
   5     8      0      1      0      12
   1     0      5      0      0       6
   1     1      0     11      0      11
   2     0      0      0      2       4

   9     6      1      2      2
     OVLP     ENTRP      GAIN      BALANC
       10    1.2208    0.1110     0.5475
        8    1.2489    0.1388
 TO (S)EPARATE, (L)UMP, (G)ENERATE, (D)ISCARD,
    (B)ACK-TRACK, (I)NITIAL OR S(T)OP ?
?G12
 ( 1 )( 2 )( 3 )( 4 )( 5 )( 6 )
  16     1      0      1      0      2      17
   1     6      0      0      0      1       7
   0     0      5      0      0      0       5
   1     0      0     11      0      0      11
   0     0      0      0      2      0       2
   2     1      0      0      0      4       7

   4     2      0      1      0      3
     OVLP     ENTRP      GAIN      BALANC
        5    1.4978    0.2496     0.5848
        8    1.2489    0.1388
 TO (S)EPARATE, (L)UMP, (G)ENERATE, (D)ISCARD,
    (B)ACK-TRACK, (I)NITIAL OR S(T)OP ?
?D6
 ( 1 )( 2 )( 3 )( 4 )( 5 )
  16     1      0      1      5      19
   1     6      0      0      0       7
   0     0      6      0      0       6
   1     0      0     11      0      11
   5     0      0      0      5       8

   7     1      0      1      5
     OVLP     ENTRP      GAIN      BALANC
        7    1.3680    0.1710     0.4615
        6    1.3877    0.1982
 TO (S)EPARATE, (L)UMP, (G)ENERATE, (D)ISCARD,
    (B)ACK-TRACK, (I)NITIAL OR S(T)OP ?
?L15
```

Fig. 3.6 ISOETRP run example

```
 ( 1 )( 2 )( 3 )( 4 )
  17     3     1     0      19
   3     9     1     0      11
   1     1     7     0       8
   0     0     0    11      11

   4     4     2     0
     OVLP     ENTRP       GAIN      BALANC
        5    1.2258     0.2043     0.3335
        5    1.2258     0.2043
  TO (S)EPARATE, (L)UMP, (G)ENERATE, (D)ISCARD,
     (B)ACK-TRACK, (I)NITIAL OR S(T)OP ?
?G12
 ( 1 )( 2 )( 3 )( 4 )( 5 )
   7     2     0     1     0      10
   2     8     0     1     2      10
   0     0     7     0     0       7
   1     1     0    10     0      10
   0     2     0     0    12      13

   3     5     0     2     2
     OVLP     ENTRP       GAIN      BALANC
        6    1.4668     0.2095     0.1897
        6    1.4668     0.2095
  TO (S)EPARATE, (L)UMP, (G)ENERATE, (D)ISCARD,
     (B)ACK-TRACK, (I)NITIAL OR S(T)OP ?
?L12
 ( 1 )( 2 )( 3 )( 4 )
  12     0     1     0      13
   0     8     0     1       8
   1     0     9     0       9
   0     1     0    15      16

   1     1     1     1
     OVLP     ENTRP       GAIN      BALANC
        2    1.3119     0.4373     0.2784
        2    1.3122     0.4374
  TO (S)EPARATE, (L)UMP, (G)ENERATE, (D)ISCARD,
     (B)ACK-TRACK, (I)NITIAL OR S(T)OP ?
?G1234
 ( 1 )( 2 )( 3 )( 4 )( 5 )( 6 )
  10     0     0     0     0     0      10
   0     3     0     1     0     0       4
   0     0     7     0     0     0       7
   0     1     0     8     0     0       8
   0     0     0     0    12     0      12
   0     0     0     0     0     4       4

   0     1     0     1     0     0
     OVLP     ENTRP       GAIN      BALANC
        1    1.6750     0.8375     0.3906
        2    1.3122     0.4374
  TO (S)EPARATE, (L)UMP, (G)ENERATE, (D)ISCARD,
     (B)ACK-TRACK, (I)NITIAL OR S(T)OP ?
?D2
```

Fig. 3.6 Cont'd

```
 ( 1 )( 2 )( 3 )( 4 )( 5 )
  12     0     1     0     0       13
   0     7     0     0     0        7
   1     0     9     0     0        9
   0     0     0    12     0       12
   0     0     0     0     4        4

   1     0     1     0     0
      OVLP      ENTRP       GAIN      BALANC
         1     1.5219     0.7610      0.3651
         1     1.5219     0.7610
  TO (S)EPARATE, (L)UMP, (G)ENERATE, (D)ISCARD,
     (B)ACK-TRACK, (I)NITIAL OR S(T)OP ?
?D5
 ( 1 )( 2 )( 3 )( 4 )
  12     0     1     0       13
   0     8     0     1        8
   1     0     9     0        9
   0     1     0    15       16

   1     1     1     1
      OVLP      ENTRP       GAIN      BALANC
         2     1.3119     0.4373      0.2784
         1     1.5219     0.7610
  TO (S)EPARATE, (L)UMP, (G)ENERATE, (D)ISCARD,
     (B)ACK-TRACK, (I)NITIAL OR S(T)OP ?
?B1
 ( 1 )( 2 )( 3 )( 4 )( 5 )
  12     0     1     0     0       13
   0     7     0     0     0        7
   1     0     9     0     0        9
   0     0     0    12     0       12
   0     0     0     0     4        4

   1     0     1     0     0
      OVLP      ENTRP       GAIN      BALANC
         1     1.5219     0.7610      0.3651
         1     1.5219     0.7610
  TO (S)EPARATE, (L)UMP, (G)ENERATE, (D)ISCARD,
     (B)ACK-TRACK, (I)NITIAL OR S(T)OP ?
?T
```

Fig. 3.6 Cont'd

?D6 ---- discard the 6th cluster,

?L15 --- lump the cluster pair 1 and 5 into one cluster,

?G12 --- create a new one from 1 and 2,

?L12 --- lump 1 and 2 into one,

?G1234 - create two new clusters from pairs 1 and 2, 3 and 4,

?D2 ---- discard the 2nd cluster,

?D5 ---- discard the 5th cluster,

?B1 backtrack the previous result,

?T ----- stop.

The final result is

OVLP = 1,

ENTRP = 1.5219,

GAIN = 0.7610,

BALANCE = 0.3651,

which is much better than the initial result. This demonstrates that the overlap table is very helpful and the algorithm very effective.

## 3.5. Simulation Results

Simulation of the first version of ISOETRP was done [3.8], where 100 classes of randomly generated data were used. Each class is assumed to have deviation values

$$D_x = 0.017, \quad D_y = 0.016.$$

ISOETRP and other 10 clustering algorithms, which were MAXIMIN, KMEANS , ISODATA (see refs. [3.1, 3.6]), NEAREST NEIGHBOR, FARTHEST NEIGHBOR, GROUP AVERAGE, CENTROID, MEADIAN, WARD'S and FLEXIBLE (see refs. [3.4, 3.7]), were tested on these data. A comparison of their performances is given in Table 3.1.

From these results we can see : entropy reduction by ISOETRP is generally good; OVLP and GAIN by ISOETRP are better than those obtained by the other algorithms except CENTROID; ISOETRP is much better than all other algorithms in BALANCE. However the first version of ISOETRP was still not very satisfactory. Althuogh it has been applied to Chinese character recognition in [3.11], where a recognition rate of 98.5% was obtained, it needs improvement, which led to the development of its interactive version.

The interactive version of ISOETRP was compared with KMEANS and ISOETRP using a 4-D data of 806 Chinese characters. Each class, or character category, contains 10 noisy samples. The vertical and horizontal profiles of each

Table 3.1 Comparison of ISOETRP with other clustering algorithms

| algorithms | no. of classes in each cluster. | | | | | | | | | | OVLP | ENTRP | GAIN | BLNC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ISOETRP | 14 | 13 | 11 | 8 | 13 | 14 | 14 | 14 | 10 | | 11 | 2.33 | 0.212 | 2.18 |
| MAXMIN | 14 | 11 | 12 | 10 | 20 | 11 | 8 | 10 | 6 | 10 | 12 | 2.39 | 0.199 | 3.77 |
| K-MEANS | 13 | 8 | 6 | 13 | 7 | 14 | 45 | 13 | 13 | 10 | 12 | 2.41 | 0.200 | 3.19 |
| ISODATA | 9 | 7 | 12 | 12 | 11 | 9 | 20 | 10 | 12 | 11 | 13 | 2.42 | 0.186 | 3.47 |
| NEAREST-NEIGHBOR | 17 | 8 | 13 | 27 | 23 | 10 | 9 | 8 | 6 | 10 | 31 | 2.54 | 0.082 | 7.03 |
| FARTHEST-NEIGHBOR | 11 | 6 | 7 | 15 | 17 | 7 | 19 | 10 | 11 | 11 | 14 | 2.40 | 0.172 | 4.38 |
| CENTROID | 11 | 13 | 13 | 23 | 9 | 7 | 7 | 10 | 6 | 11 | 10 | 2.35 | 0.235 | 4.88 |
| MEADIAN | 11 | 11 | 7 | 14 | 16 | 8 | 20 | 21 | 9 | 5 | 22 | 2.49 | 0.113 | 5.43 |
| WARD'S | 12 | 13 | 13 | 11 | 10 | 10 | 11 | 11 | 18 | 10 | 19 | 2.51 | 0.132 | 2.42 |
| FLEXIBLE | 12 | 17 | 13 | 12 | 9 | 6 | 11 | 19 | 11 | 8 | 18 | 2.47 | 0.137 | 3.91 |
| GROUP-AVERAGE | 12 | 17 | 13 | 23 | 10 | 7 | 7 | 9 | 5 | 11 | 14 | 2.39 | 0.170 | 5.34 |

sample were obtained, to which the fast Walsh transform was applied to obtain 64 numerical features. For a given number of classes, 4 best features were selected according to Fisher's criterion. In the 4-D space, each character category occupies a hyper ellipsoid containing the 10 samples. When running the KMEANS program, the number of clusters was predefined but different values were tried to get the best possible results based on the GAIN. These results were then used as the initial states of the ISODATA algorithm. When running ISODATA some parameters were adjusted in an attempt to obtain a good value of GAIN.

The experience we gained from using the ISOETRP algorithm can be summarized as follows :

DIVIDE, DROP, LUMP and CREATE are all effective. A good strategy is to use DIVIDE and CREATE alternatively until more than enough clusters have been created, then apply LUMP and DROP alternatively to prune the clusters. Usually good results are obtained and stored in RLTS in this process. RETRIEVE and INITIAL are necessary some times, and the former is more important than the latter.

A comparison of these 3 algorithms is presented in Table 2-6, for cases n = 50, 100, 200, 400 and 806 respectively, where BLANC, ENTRP, GAIN and PIDX have been defined in Sections 1 and 2. On examination of this table, one can see the following results :

Table 3.2 Comparison of ISOETRP, KMEANS and ISODATA

No. of Samples: 50

| | No. of samples in Clusters | BALNC | ENTRP | OVLP | GAIN | | PIDX | |
|---|---|---|---|---|---|---|---|---|
| ISOETRP | 13,13,12,12 | 0.0400 | 1.3855 | 0 | 2.7710 | 3.4379 | 84.45 | 56.7 |
| | 13,12,11,7,7 | 0.2530 | 1.5764 | 0 | 3.1527 | | 59.96 | |
| | 12,9,9,7,7,6 | 0.2366 | 1.7648 | 0 | 3.5296 | | 52.59 | |
| | 10,9,9,7,7,6,2 | 0.3466 | 1.8729 | 0 | 3.7458 | | 44.95 | |
| | 9,9,7,7,7,6,3,2 | 0.3816 | 1.9951 | 0 | 3.9902 | | 41.82 | |
| KMEANS | 15,15,13,8 | 0.2244 | 1.3292 | 1 | 0.8861 | 1.5345 | 80.66 | 57.9 |
| | 18,14,13,8,7 | 0.2623 | 1.5255 | 2 | 0.6102 | | 63.46 | |
| | 12,9,9,9,8,6 | 0.2006 | 1.7143 | 3 | 0.4898 | | 55.37 | |
| | 10,9,9,7,7,6,2 | 0.3464 | 1.8729 | 0 | 3.7458 | | 44.95 | |
| | 12,9,8,6,5,5,3,3 | 0.4569 | 1.9407 | 1 | 1.9407 | | 45.07 | |
| ISODATA | 13,12,11,7,7 | 0.2530 | 1.5764 | 0 | 3.1527 | 1.1088 | 60.79 | 57.7 |
| | 14,13,9,9,7 | 0.2551 | 1.5307 | 2 | 0.6123 | | 64.64 | |
| | 14,13,12,8,6,4 | 0.3927 | 1.5332 | 7 | 0.2044 | | 65.62 | |
| | 14,11,8,6,6,5,4 | 0.4311 | 1.7641 | 4 | 0.3920 | | 49.91 | |
| | 13,11,10,7,4,4,2 | 0.5276 | 1.7740 | 1 | 1.1827 | | 47.86 | |

Table 3.3 Comparison of ISOETRP, KMEANS and ISODATA

No. of Samples: 100

| | No. of Samples in Clusters | BALNC | ENTRP | OVLP | GAIN | | PIDX | |
|---|---|---|---|---|---|---|---|---|
| ISOETRP | 16,16,16,14,13,12,8,7 | 0.2623 | 2.0261 | 2 | 0.8104 | 0.5474 | 81.26 | 142.27 |
| | 64,37 | 0.2673 | 1.0311 | 1 | 0.4290 | | 251.70 | |
| | 43,25,24,10 | 0.4954 | 1.2599 | 2 | 0.5040 | | 154.61 | |
| | 17,16,16,14,14,12,8,7 | 0.2692 | 2.0074 | 4 | 0.4461 | | 81.52 | |
| KMEANS | 47,33,28 | 0.2234 | 1.0045 | 8 | 0.1182 | 0.1877 | 188.9 | 145.01 |
| | 42,30,17,16 | 0.4053 | 1.2665 | 5 | 0.2303 | | 158.19 | |
| | 43,22,18,17,9 | 0.5233 | 1.4127 | 9 | 0.1487 | | 125.31 | |
| | 28,22,22,17,9,8 | 0.4090 | 1.6491 | 6 | 0.2537 | | 107.63 | |
| ISODATA | 24,15,13,12,12,11,7,6,5,3 | 0.5332 | 2.0749 | 8 | 0.2441 | 0.3394 | 74.64 | 71.19 |
| | 24,15,13,12,11,11,7,6,5,3 | 0.5370 | 2.0758 | 7 | 0.2768 | | 74.73 | |
| | 16,15,15,13,11,10,7,6,5,5 | 0.4004 | 2.1937 | 3 | 0.6268 | | 69.34 | |
| | 23,13,12,11,11,9,9,6,6,6,4 | 0.4954 | 2.2029 | 10 | 0.2098 | | 66.06 | |

Table 3.4 Comparison of ISOETRP, KMEANS and ISODATA

No. of Samples: approx. 200

| | No. of Samples in Clusters | BALNC | ENTRP | OVLP | GAIN | | PIDX | |
|---|---|---|---|---|---|---|---|---|
| ISOETRP | 68,49,43,21,18,12 | 0.5643 | 1.5754 | 11 | 0.1370 | 0.1322 | 243.4 | 406.5 |
| | 123,82 | 0.2000 | 0.6503 | 5 | 0.1182 | | 492.8 | |
| | 129,75 | 0.2647 | 0.6365 | 4 | 0.1414 | | 483.2 | |
| KMEANS | 76,72,63 | 0.0773 | 1.0409 | 11 | 0.0905 | 0.0707 | 362.0 | 311.1 |
| | 60,59,54,51 | 0.0656 | 1.2714 | 24 | 0.0519 | | 301.9 | |
| | 52,48,45,40,36 | 0.1283 | 1.5016 | 21 | 0.0698 | | 269.4 | |
| ISODATA | 41,40,39,33 30,22,13,12 | 0.3481 | 1.8567 | 30 | 0.0609 | 0.0792 | 189.0 | 189.7 |
| | 64,47,28,26 18,14,11,11 | 0.6499 | 1.8184 | 19 | 0.0932 | | 196.4 | |
| | 45,42,35,33 23,21,12,11 | 0.4372 | 1.8761 | 22 | 0.0834 | | 183.6 | |

Table3.5 Comparison of ISOETRP, KMEANS and ISODATA

No. of Samples: 400

| | No. of Samples in Clusters. | BLANC | ENTRP | OVLP | GAIN | | PIDX | |
|---|---|---|---|---|---|---|---|---|
| ISOETRP | 157,140,123 | 0.0991 | 1.0450 | 20 | 0.0510 | 0.0501 | 747.6 | 747.2 |
| | 160,140,120 | 0.1166 | 1.0441 | 20 | 0.0509 | | 747.4 | |
| | 159,139,123 | 0.1049 | 1.0420 | 21 | 0.0485 | | 746.7 | |
| KMEANS | 164,137,122 | 0.1232 | 1.0368 | 23 | 0.0441 | 0.0416 | 746.5 | 629.3 |
| | 142,126,85,78 | 0.2503 | 1.2756 | 31 | 0.0405 | | 605.4 | |
| | 123,118,72,63,60 | 0.3156 | 1.4703 | 36 | 0.0403 | | 535.9 | |
| ISODATA | 112,100,84,64,62,28 | 0.3681 | 1.5979 | 50 | 0.0316 | 0.0305 | 492.1 | 487.4 |
| | 119,108,73,61,59,33 | 0.3916 | 1.5872 | 53 | 0.0293 | | 485.2 | |
| | 117,113,70,62,59,31 | 0.4053 | 1.5781 | 52 | 0.0301 | | 484.9 | |

Table 3.6 Comparison of ISOETRP, KMEANS and ISODATA

No. of Samples: 806

| | No. of Samples in Clusters | BALNC | ENTRP | OVLP | GAIN | | PIDX | |
|---|---|---|---|---|---|---|---|---|
| ISOETRP | 345,293,223 | 0.1743 | 1.0162 | 55 | 0.0183 | 0.0184 | 1550.5 | 1550.8 |
| | 348,271,242 | 0.1558 | 1.0205 | 55 | 0.0184 | | 1543.7 | |
| | 342,308,210 | 0.1952 | 1.0127 | 54 | 0.0186 | | 1558.2 | |
| KMEANS | 328,279,273 | 0.0840 | 1.0074 | 74 | 0.0135 | 0.0139 | 1627.8 | 1364.8 |
| | 334,212,176,173 | 0.2926 | 1.2303 | 89 | 0.0137 | | 1359.0 | |
| | 293,168,164,143,133 | 0.3284 | 1.4407 | 99 | 0.0145 | | 1107.6 | |
| ISODATA | 326,312,243 | 0.1253 | 1.0007 | 74 | 0.0134 | 0.0128 | 1576.5 | 1199.1 |
| | 248,175,152 148,109,96 | 0.3201 | 1.6055 | 122 | 0.0131 | | 1027.0 | |
| | 278,157,135, 114,95,86,80 | 0.4718 | 1.6753 | 139 | 0.0120 | | 993.9 | |

a) for BLANC and ENTRP, ISOETRP is generally better than KMEANS and ISODATA.

b) For OVLP, ISOETRP is obviously better than the other two, because the operator can see the cluster structure and control the overlaps. There is no mechanism to control overlap in the other two (or any other) clustering algorithms.

c) for GAIN, ISOETRP is absolutely superior to the others, since this is the overall objective of this algorithm.

d) KMEANS and ISODATA are good and popular clustering algorithms. But they aim at the minimization of PIDX (eqn.3.1), which is quite different from GAIN. In the table, it can be seen that ISOETRP is not as good as KMEANS and ISODATA with respect to PIDX. Also the performance index value of ISODATA is better than KMEANS. However the reverse is true for GAIN. In the measure PIDX, each sample is considered purely as a point. While in GAIN, each sample is a class occupying a "sample region" in the higher dimensional space. GAIN measures the ratio of entropy reduction over overlap, which does not exist among the "points". Since ISODATA and KMEANS are not conscious of "sample regions", they do not optimize GAIN.

With respect to computing cost, we notice that ISOETRP needs more time than KMEANS and ISODATA to arrive at such good results. If ISODATA takes time T in one run, (excluding the time used to try different parameters), then ISOETRP takes about 3T. The reason is that ISOETRP has much more complex objectives and hence more computations are necessary in order to achieve these objectives. Anyhow, none of them takes too much time. For example, in the case of 806 classes we tested, ISOETRP used 2-3 minutes of the CPU time (on a Cyber-172 computer), which represents a worthwhile one-shot investment in the design phase of the tree classifier.

## 3.6. Conclusion

A new clustering algorithm ISOETRP with totally new objectives has been proposed based on those theoretical results in Chapter 2, which can be directly applied to design the tree classifier. The success of the entropy reduction concept in this algorithm leads to its potential in the pattern recognition field. Overlap, which was once considered as an open problem in clustering analysis, has been treated. It is interesting that overlap even gives us some information about the structure of the given data via the newly proposed overlap table. This constitutes the main reason of the effectiveness and ease of the interactive version of ISOETRP, which shows better performance in optimizing the GAIN than other well-known clustering algorithms.

## 3.7. References


[1]. G. H. Ball and D. J. Hall, "ISODATA, a novel method of data analysis and pattern classification," NTIS Report, AD699616, 1965.

[2]. O. Kakusho and R. Mizoguchi, "A new nonlinear mapping algorithm and its application to dimension and cluster analysis," Proc. of the 5th Conf. on Pattern Recognition, pp. 430-432, Dec. 1980.

[3]. J. B. Kruskal, "Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis," Psychometrika, vol.29, p.1, 1964.

[4]. John A. Hartigon Clustering Algorithms, John Wily & Sons, Inc., New York, Toronto, 1975.

[5]. L. W. Johnson and R. D. Riess, Numerical Analysis, Chapter 3, Addison-Wesley, 1977.

[6]. J. T. Tou and R. C. Gonzalez, Pattern Recognition Principles, Chapter 3, pp.92-103, Addison Wesley, London, Amsterdam, Don Mill, Sydney, Tokyo, 1974.

[7]. Michael R. Anderberg, Cluster Analysis for Applications, Chpter 6, Academic Press, New York, 1973.

[8]. Q. R. Wang and C. Y. Suen, "ISOETRP -- A new clustering algorithm," Technical Report, Dept. of Computer Science, Concordia University, Dec. 1981.

[9]. C. Y. Suen and Q. R. Wang, "ISOETRP -- An interactive clustering algorithm with new objectives," in press, Pattern Recognition.

[10]. E. Diday and J. C. Simon, "Clustering analysis," in Digital Pattern Recognition, K. S. Fu (ed.), pp. 47-94, Springer-Verlag, Berlin, Heidelberg, New York, 1976.

[11]. Q. R. Wang, Y. X. Gu and C. Y. Suen, "A preliminary study on computer recognition of Chinese characters printed in different fonts," Proc. Int. Conf. of the Chinese Language Computer Society, 344-351, Sept. 1982.

# CHAPTER FOUR

# FEATURE ANALYSIS

## INTRODUCTION

Different from the case of a matching scheme, features in a large classifier must be computed once for all levels of classification. This is the reason that FFT or fast Walsh transform are more suitable than the Karhunen-Loève transform. Global features, such as profiles and mesh, were used in the tree classifier simulation. Section 2 gives some analysis and experimental results of the profiles of Chinese characters. Section 3 covers the original work on phase features, which were proposed and analyzed for pattern recognition for the first time in [1.16]. There are two feature measures, information content measure and Fisher's criterion, suitable for feature selection in large class problems. To obtain a good understanding of the relationship between these two measures, a tentative deduction was made under some assumption in [4.23], which is described in Section 4.

## 4.1. Orthonormal Transformation in Feature Extraction

As has been seen in Section 1.2, K-L transform is a powerful tool in feature extraction in the matching scheme. On the other hand, other transforms are more suitable for a tree classifier. As introduced in Section 1.2, K-L transform needs computation of some $\Phi_j$'s in eqn. (1.4), each of which is a matrix with the same dimension size of the original pattern. They vary from pattern class to pattern class, and have to be computed for each class individually. If K-L transform were used, it would have to be computed at each internal node of the tree in the design phase. Furthermore much computation would be necessary for the unknown pattern to pass through each level of the tree in the recognition phase, which is practically impossible in a large tree classifier. In addition, the information about each concrete K-L transform must be stored in the internal node, which would take up much more memory than the data structure described in Section 1.3, and cause too much trouble in the tree implementation. The other transforms, such as Haar, Walsh and FFT [4.11], can be specified once for all the internal nodes. Only a "global" computation of these transforms with addition of "local indexing" (see Algorithm 5.1 in Chapter 5) are necessary in both design and recognition phases. In this way, the tree needs far less memory, and is much easier to represent than using the K-L

transform. The recognition speed is much faster than the K-L case. This explains why Harr, Walsh and Fourier transforms serve the large tree classifier much better than the K-L transform.

There are lots of studies in orthonormal transforms and K-L transform is considered the optimal one in data compression. In [4.12], a comparison of K-L, FFT, Walsh and Harr transform on the raw data, and the row data itself was given. The experiments showed that their performances rank in the same order. Nevertheless, FFT and Walsh do behave very well.

Both FFT and fast Walsh transforms can be computed much faster than K-L transform. Because of their uniform formulas, hardware chips are even available, which gives a substantial saving in feature extraction cost. In addition, it is well known that fast Walsh transform has the advantage of lower cost than FFT.

The comparison in [4.12] indicates that FFT seems to be better than Walsh transform in data compression. But for binary images, such as binarized character patterns, FFT is not necessarily better. The reason is that Gibbs phenomenon occurs in Fourier expansion when the original function is not continuous, and this condition holds in our character matrix. In [4.21], many results about convergence properties of Walsh transform were given, some of which do

not have correspondence in Fourier transform. But to what extent these factors affect feature extraction can only be answered by experiments. Such an experiment was conducted in [1.14], where both Fourier and Walsh transforms were applied on the projection profiles (see Section 2) of 200 Chinese characters to obtain $F_1$ - $F_{64}$ and $W_1$ - $W_{64}$ features for comparison. $F_1$ - $F_{64}$ are actually amplitude features derived from Fourier transforms, as used in [4.1-4.4]. They are listed in descending order of the feature measure $J_1'$ (see Section 4) in Table 4.1. We can see from this table that Walsh coefficients are generally better than the Fourier's. Based on the above consideration, both Walsh transform and Fourier transform were used in the tree classifier simulation. The improvement of FFT feature extraction is the introduction of phase features with plenty of analysis, which will be given in Section 3.

Table 4.1 Feature ordering: F's and W's

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $W_9$ | $W_7$ | $W_{13}$ | $F_6$ | $T_B$ | $W_{11}$ | $F_5$ | $W_3$ | $W_5$ | $F_3$ | $F_7$ | $W_{15}$ | $F_4$ |
| $F_8$ | $F_{35}$ | $W_{19}$ | $W_{17}$ | $W_{21}$ | $F_{36}$ | $W_{25}$ | $W_4$ | $W_{14}$ | $W_6$ | $F_9$ | $W_8$ | |
| $W_{12}$ | $W_{18}$ | $W_{16}$ | $F_2$ | $W_{23}$ | $W_{29}$ | $W_{47}$ | $W_{20}$ | $F_{39}$ | $W_{24}$ | $W_{51}$ | $W_{10}$ | |
| $W_{27}$ | $F_{37}$ | $W_{22}$ | $W_{49}$ | $W_{55}$ | $F_{10}$ | $W_{53}$ | $W_{28}$ | $W_{57}$ | $W_{26}$ | $F_{40}$ | $W_{45}$ | |
| $W_{30}$ | $W_{31}$ | $W_{46}$ | $W_{36}$ | $W_{33}$ | $F_{42}$ | $F_{43}$ | $W_{42}$ | $F_{38}$ | $W_{43}$ | $W_{35}$ | $W_{59}$ | |
| $W_{40}$ | $W_{50}$ | $F_{45}$ | $F_{46}$ | $W_{32}$ | $W_{37}$ | $W_{48}$ | $W_{61}$ | $W_{44}$ | $W_{54}$ | $F_{44}$ | $W_{34}$ | |
| $W_{38}$ | $W_{52}$ | $F_{11}$ | $F_{31}$ | $W_{58}$ | $F_{47}$ | $W_{56}$ | $F_{20}$ | $F_{38}$ | $F_{36}$ | $W_{62}$ | $W_{60}$ | |
| $F_{17}$ | $W_{39}$ | $W_{41}$ | $W_{63}$ | $F_{48}$ | $W_{64}$ | $F_{49}$ | $F_{54}$ | $F_{15}$ | $F_{50}$ | $F_{55}$ | $F_{52}$ | |
| $F_{16}$ | $F_{12}$ | $F_{53}$ | $F_{21}$ | $F_{51}$ | $F_{57}$ | $F_{22}$ | $F_{30}$ | $F_{33}$ | $F_{32}$ | $F_{31}$ | $F_{29}$ | |
| $F_{27}$ | $F_{56}$ | $F_{60}$ | $F_{23}$ | $F_{39}$ | $F_{61}$ | $F_{65}$ | $F_{58}$ | $F_{64}$ | $F_{63}$ | $F_{62}$ | $F_{28}$ | |
| $F_{26}$ | $F_{66}$ | $F_{14}$ | $F_{24}$ | $F_{25}$ | $F_{13}$ | | | | | | | |

(Note that $T_B$ is the total number of black points, which is identical to $F_0$, $W_1$, and $W_{65}$.

## 4.2. S.V.D. Analysis and Profiles of Chinese Characters

The projection profile as a global feature in Chinese character recogntion was first proposed in 1968 (see [4.1]) and implemented in 1972 [4.2], where Fourier transforms of vertical and horizontal profiles were used. Since then some improvements have been made [4.3, 4.4]. Profile was even used for handwritten Chinese characters associated with some other local features [4.20]. It should be pointed out that the profile of a character is sensitive to stroke width variation [4.4], rendering it unsuitable for a multifont character classifier. Nevertheless it does have the advantages of considerable data compression, stability in random noise, and ease in computation. It was used once in a tree classifier of 3155 single font Chinese characters with high recognition rate [1.14]. Also the profile of shade and crossing count were used in multifont Chinese character recogniton [3.11]. This section covers the original work [4.5], where SVD (Singular Value Decomposition of matrix) analysis of and experiments on the 4-directional profiles of Chinese characters were made. It was shown that 4-directional profiles contain much information for recognition, and in some cases they are even good for crude reconstruction of Chinese characters.

SVD stands for Singular Value Decomposition, which was called "general inverse of matrix" when proposed by Moore in 1920 for the first time. It is also known as pseudo-inverse

after studied by Penrose and others in detail [4.8]. It is introduced as follows [4.6, 4.9].

For any N x M matrix A, we can define two symmetric matrices

$$P_1 = AA^t$$

$$P_2 = A^tA$$

where $A^t$ is the transpose of A. The SVD theory says :

A) $P_1$ and $P_2$ have the same non-zero eigen values

$$\lambda_1 \geq \lambda_2 \geq \ldots\ldots \geq \lambda_r > 0 \qquad (4.1)$$

where r is the rank of A, $r \leq M$ and $r \leq N$.

B) If $u_i$ is an eigen vector of $P_1$ corresponding to $\lambda_i$, then

$$v_i = (\lambda)^{-1/2} A^t u_i \qquad (4.2)$$

is the eigen vector of $P_2$ corresponding to $\lambda_i$, (i=1, 2, ..., r) and

$$v_i^t v_j = u_i^t u_j$$

$$= \delta_{ij} = 0; \; I \neq j$$

$$1, \; I = j, \; i,j=1,2,\ldots,r \qquad (4.3)$$

C) A has the representation

$$A = \sum_{i=1}^{r} \lambda_i^{1/2} u_i v_i^t \qquad (4.4)$$

$\lambda_i$ is called the singular value of A and (4.4) is the SVD of A. If we denote the matrices U and V by the following equations:

$$U = (u_1, u_2, \ldots, u_r),$$

$$V = (v_1, v_2, \ldots, v_r),$$

then (4.4) has an alternate form

$$A = U \, \mathrm{diag}(\lambda^1, \ldots, \lambda^r, 0, \ldots, 0) \, V^t \qquad (4.5)$$

where diag(...) is the diagonal matrix with the diagonal elements denoted in the brackets.

D) if A is a matrix expression of a picture the power of which is defined as

$$\|A\| = \mathrm{Trace}\,(A^t A)$$

then

$$\|A\|^2 = \sum_{i=1}^{k} \lambda_i \qquad (4.6)$$

If $\lambda_1, \ldots, \lambda_k$ are much larger than the other singular values, then

$$A_k = \sum_{i=1}^{k} \lambda_i^{1/2} u_i v_i^t$$

is a good approach to A, i.e. information in A is compressed in $A_k$.

The last property has been applied to image processing [4.10] or character representation [4.7]. Usually the value of k depends on the nature of the picture. For a Chinese character, A is a binary matrix containing mainly vertical, horizontal and diagonal strokes, leading to a small value of k. This can be stated precisely in the following theorems.

THEOREM 4.1. If A has only vertical (SV) and horizontal (SH) strokes and the total number of strokes is n, then

$$k \leq n \tag{4.7}$$

Proof.

Suppose a SH stroke has width W and length L as shown in Fig. 4.1.a, (which may be broken somewhere as denoted by dash lines). The ith stroke can be represented as a matrix of rank 1,

$$A_i = u_i v_i^t \tag{4.8}$$

where

$u_i^t = (0, \ldots, 0, 1, \ldots, 1, 0, \ldots, 0)$ with W 1's

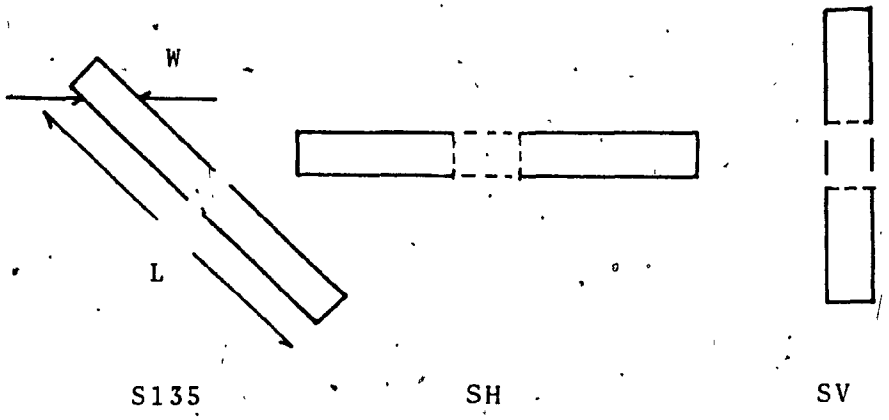
and

$v_i^t = (0, \ldots, 0, 1, \ldots, 1, 0, \ldots, 0)$ with L 1's

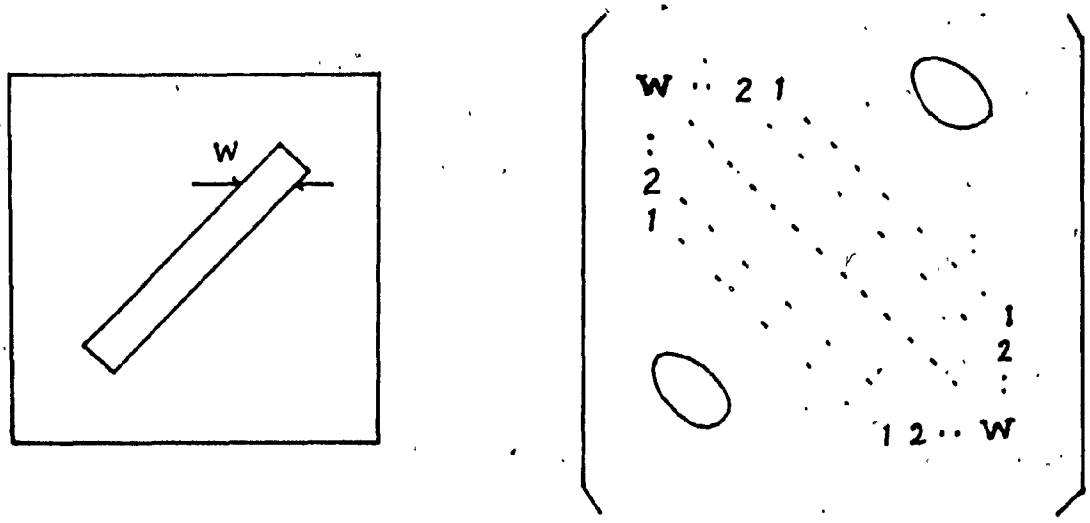
A character with only SH and SV is then expressed as the sum of some matrices of rank 1,

$$A = \sum_{i=1}^{n} A_i = \sum_{i=1}^{n} u_i v_i^t$$

the rank of which, r, is certainly not larger than n.




a) Stroke definition




b) A $45^0$ Stroke and its $AA^t$

Fig. 4.1 Strokes

This proves (4.7).

QED.

This result guarantees that if a character has only SH and SV strokes, then the power of the matrix is condensed within only a few terms of the SVD. Many Chinese characters mainly contain SH and SV strokes. For paralell strokes, which is usually the case, the rank of A will be much smaller than the number of strokes. Fig.4.2 shows such examples, where the rank r is much smaller than the number of strokes, n.

For diagonal strokes, we can prove the following theorem:

THEOREM 4.2.

If A has only diagonal strokes (i.e. in $45^0$ or $135^0$, denoted by S45 and S135 respectively), then the largest singular value

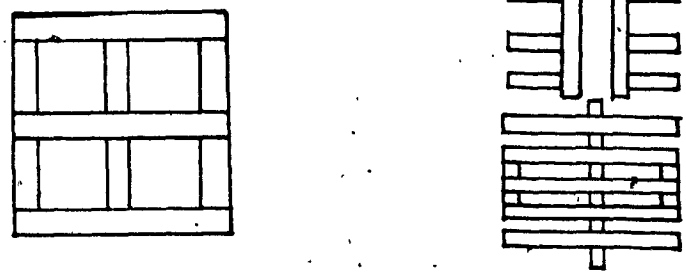
$$\lambda_1^{1/2} \leq n\ W \qquad (4.9)$$

where W is the width of the strokes.

Proof.

Let us use induction on the number of strokes, n.

When n = 1, consider Fig.4.1, where the matrix $AA^t$ is represented in b), W being the width of the stroke in



n = 6, r = 2 n = 16, r = 5

Fig. 4.2 Examples of Pattern Matrix Rank

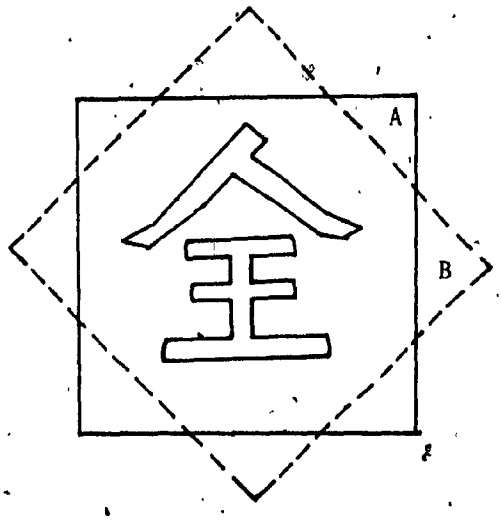



Fig. 4.3 4-view SVD of a Chinese Character

$45^0$ direction (S45). For any eigen value $\lambda_i$ of $AA^t$, suppose $u_{ij}$ is the element of the corresponding eigen vector $u_i$, which has the largest absolute value among other elements. We construct a matrix $U_i$,

$$U_i = AA^t u_i = \lambda_i u_i \tag{4.10}$$

the elements of which are denoted by $U_{i,j}$. The following is a straight forward deduction,

$$|U_{ij}| = \lambda_i |u_{ij}| \tag{4.11}$$

$$\leq W |u_{i,j}| + (W-1)(|u_{i,j-1}| + |u_{i,j+1}|) + \ldots + 1\,(|u_{i,j-W+1}| + |u_{i,j+W-1}|)$$

$$\leq (W + 2(W-1) + \ldots + 4 + 2)\,|u_{ij}|$$

$$= W^2 |u_{ij}| \tag{4.12}$$

which implies

$$\lambda_i^{1/2} \leq W, \text{ for } n = 1. \tag{4.13}$$

Now suppose eqn.(4.9) is true for n=m and let us prove it for the case n=m+1. Matrix A can be expressed as the sum of 2 matrices

$$A = A_m + A_1 \tag{4.14}$$

where $A_m$ contains m strokes and $A_1$ a single stroke, all of which are in diagonal directions. We can assume

$$A_1 = U\,\mathrm{diag}(\lambda_1^{1/2}, \lambda_2^{1/2}, \ldots, \lambda_{r_1}^{1/2})\,V^t,$$

$$\lambda_i^{1/2} \leq W, \quad i=1, 2, \ldots, r_1 \qquad (4.15)$$

$$A_m = S \operatorname{diag}(\mu_1^{1/2}, \mu_2^{1/2}, \ldots, \mu_{r_2}^{1/2})\, T^t,$$

$$\mu_j^{1/2} \leq mW, \quad j = 1, 2, \ldots, r_2 \qquad (4.16)$$

where U, V, S and T are matrices with columns or rows as the corresponding eigen vectors. For any vector u of norm 1, we have

$$\|A_1 u\| \leq W^2. \qquad (4.17)$$

in fact,

$$\|A_1 u\| = \|U \operatorname{diag}(\lambda_1^{1/2}, \lambda_2^{1/2}, \ldots, \lambda_{r_1}^{1/2})\, V^t u\|$$

by eqn. (4.15). Let

$$v = V^t u,$$

then

$$\|v\| \leq 1,$$

and we have

$$\|\operatorname{diag}(\lambda_1^{1/2}, \lambda_2^{1/2}, \ldots, \lambda_{r_1}^{1/2})\, v\| \leq W \qquad (4.18)$$

by (4.13), and (4.17) is proved.

Similarly we can prove

$$\|A_1^t u\| \leq W$$

$$\|A_m u\| \leq m W$$

$$\|A_m^t u\| \leq m W \qquad (4.19)$$

from which, it is easy to show that, for any vector of norm 1,

$$\|AA^t u\| = \|(A_m + A_1)(A_m^t + A_1^t) u\|$$

$$\leq \|A_m A_m^t u\| + \|A_1 A_1^t u\|$$

$$+ \|A_m A_1^t u\| + \|A_1 A_m^t u\|$$

$$\leq m^2 W^2 + W^2 + mWW + WmW$$

$$= (m + 1)^2 W^2$$

QED.

Considering the result of this theorem, it is very interesting to note that the length does not appear in eqn.(4.9). If a SH or SV stroke is long, then the corresponding eigen values will be large, while for S45 and S135 strokes, the eigen values do not exceed a bound $(nW)^2$, which is independent of L. The reason is that the power is spread across many eigen values and none of them will exceed the bound given in (4.9). This can be explained as follows. If a single S45 stroke of length L=50 and width W=3 is put in a 64x64 matrix, then the number of black points is N=150, and

$$\lambda_1 \leq W^2 = 9$$

by (4.9), which implies the number of significant eigen values is not less than

$$N/W^2 \simeq 17$$

which means the information of the S45 stroke is spread over at least 17 terms. For a double S45 pattern, this value is about 8, still large.

To summarize, Theorems 4.1 and 4.2 tell us that the power of SH and SV are mainly contained in the first few terms of the SVD, and the power of S45 and S135 strokes is spread over many terms. The first few terms do not contain much information of S45 and S135 strokes. Due to these results, the 4-view-SVD, which is depicted in Fig.4.3, should be better than the original SVD (we call it 2-view SVD). In Fig.4.3; the character "全" is recorded in matrices A and B, and we have by eqn.(4.4) :

$$A = \sum_{i=1}^{r_1} \lambda_i^{1/2} u_i v_i^t$$

$$B = \sum_{j=1}^{r_2} \mu_j^{1/2} x_j y_j^t \qquad (4.20)$$

where $r_1$ and $r_2$ are the ranks of A and B respectively. The power of SH and SV strokes are mainly contained in the first $k_1$ terms of the SVD of A, by Theorem 4.1. while in the SVD of B, this power is spread over many terms by Theorem 4.2. Similarly, the power of S45 and S135 strokes is mainly contained in the first $k_2$ terms in the SVD of B, but spread

over many terms of the SVD of A. If we use

$$C = \sum_{i=1}^{k_1} \lambda_i^{1/2} u_i v_i^t + \sum_{j=1}^{k_2} \mu_j^{1/2} x_j y_j^t \qquad (4.21)$$

then $k_1 + k_2$ terms will be enough to contain most information of the character.

Eqn. (4.21) has been applied to reconstruct some Chinese characters in the following steps :

1) Get matrix A.

2) Get matrix B by rotating A through $45^0$ and apply some filling and smoothing operations.

3) Calculate the SVD of A and B respectively. The first few singular values of the character "全" are shown in Table 4.2. One sees from the table that, the first few terms of each SVD contain much power, and the first term is especially larger than all the others.

4) Reconstruct the character using formula (4.21) with

$$(k_1, k_2) = (1, 0), (0, 1), (1, 1), \ldots, (5, 5)$$

respectively, the results of which are shown in Fig.4.4. It is seen from these figures that : even when $(k_1, k_2) = (1, 1)$, the reconstruction is satisfactory, which is better than $(k_1, k_2) = (2, 0)$ or $(0, 2)$. When $(k_1, k_2) = (2, 2)$ or $(5, 5)$, the reconstruction is nearly complete.

Table 4.2. The largest singular values

(matrix dimension 50X50)

| No. | A | B |
|---|---|---|
| 1 | 217.31 | 267.96 |
| 2 | 62.96 | 43.11 |
| 3 | 39.53 | 31.54 |
| 4 | 30.06 | 21.20 |
| 5 | 28.11 | 21.16 |
| 6 | 17.10 | 13.74 |
| 7 | 10.82 | 11.01 |
| 8 | 10.36 | 9.70 |

Table 4.3. COS θ, θ between profiles

and eigen vectors

| characters | | | | |
|---|---|---|---|---|
| V | 0.982 | 0.997 | 0.992 | 0.992 |
| H | 0.989 | 0.998 | 0.997 | 0.997 |
| 45 | 0.997 | 0.985 | 0.996 | 0.999 |
| 135 | 0.995 | 0.995 | 0.996 | 0.991 |

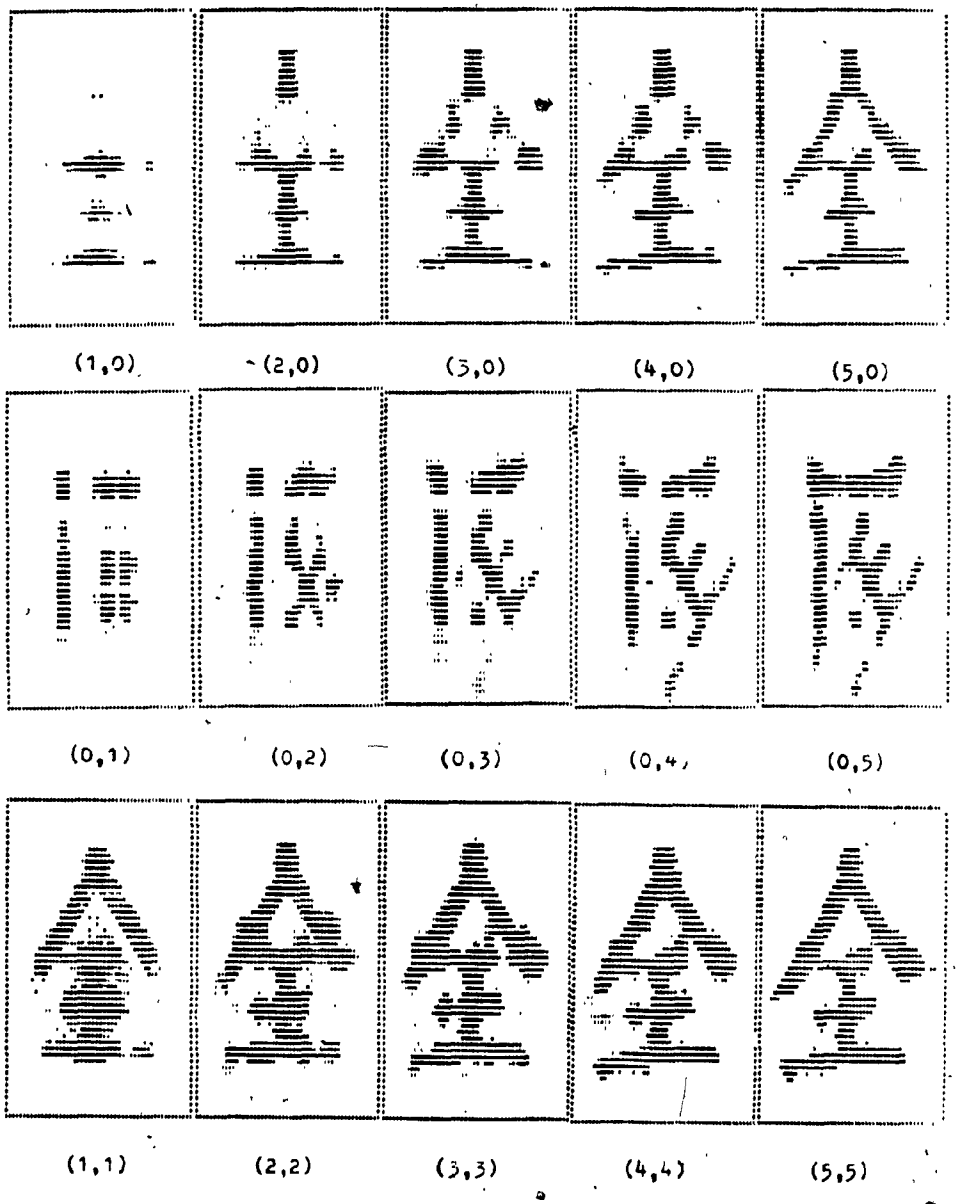



Fig. 4.4 Reconstruction using 4-view SVD.

Ref. [4.7] gives a result about the relation between profiles and the SVD, which can be summarized as follows :

PROPOSITION.

In eqn.(4.4), if $\lambda_1$ is much larger than the rest, then $u_1$ and $v_1$ will almost completely coincide with 2 profiles in the vertical and horizontal directions respectively.

Some experiments have been done on Chinese characters to verify this proposition. In our experiment, 2 diagonal directions were included. The results are shown in Table 4.3, where the 4-profiles coincide with $u_1$ and $v_1$ very well. Since the 4 eigen vectors contain much information, or power of the character, the 4-profiles should contain much information of the character. A further experiment of character reconstruction using 4-profiles was conducted on character " 周 ", " 网 ", and " 綱 ", (Fig.4.5). It is clear in these results that 4-profiles really contain much information for crude reconstruction of Chinese characters. From this we can conclude that 4-view profiles should be useful in Chinese character recognition if we use them properly.

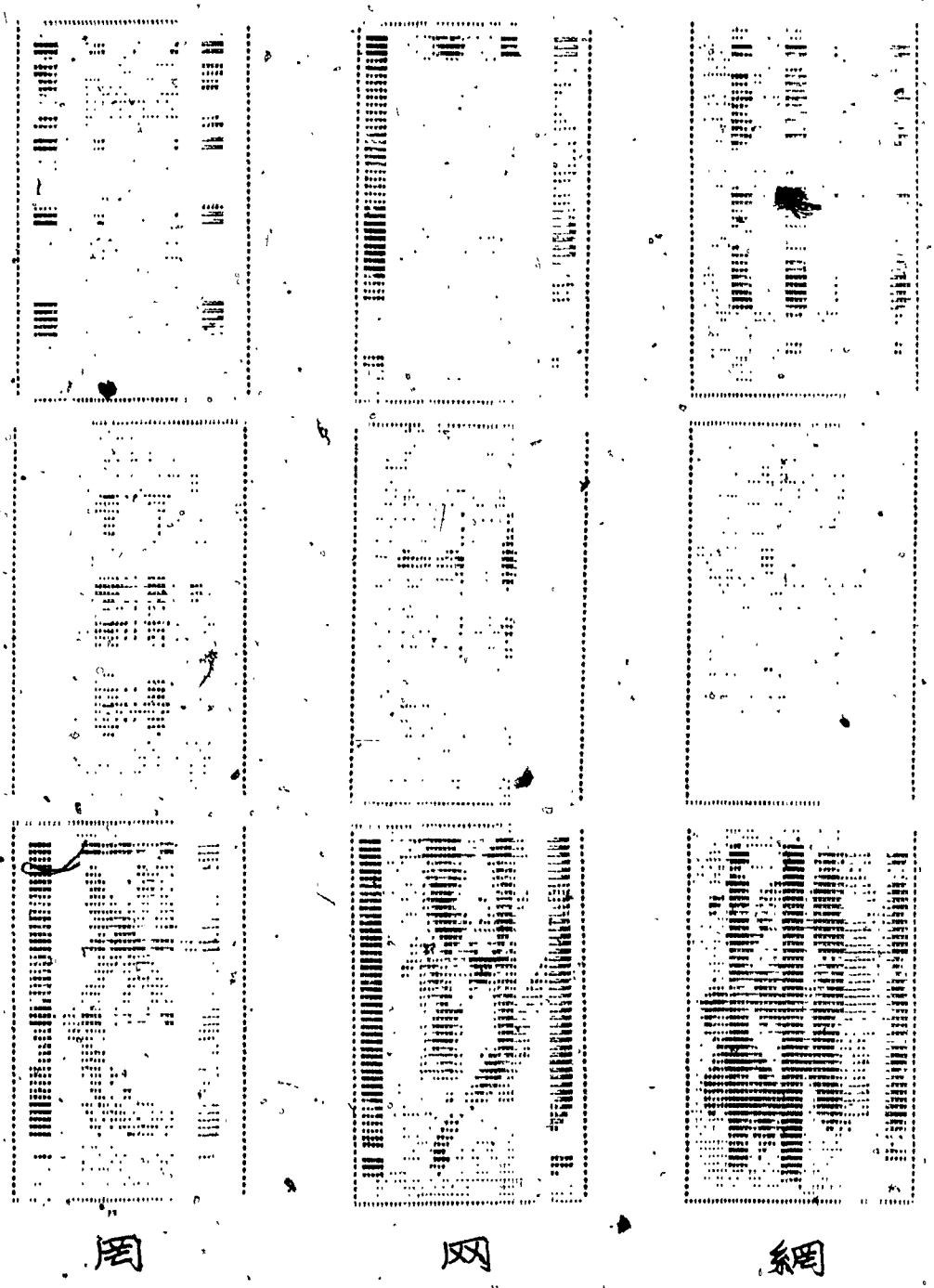


Fig. 4.5 Reconstruction using 4-profiles.

## 4.3. Phase Features

FFT has been widely used in the field of pattern recognition. In refs. [4.2, 4.4, 4.14], some analysis and experiments on feature extraction from Chinese characters have been reported. However only the magnitude features have been used :

$$M(i, j) = (Im(F_{ij})^2 + Re(F_{ij})^2)^{1/2} \qquad (4.22)$$

where $F_{ij}$ is the Fourier transform of the image matrix, Im and Re are the imaginary and real parts respectively. The image matrix may be the character matrix or some other pattern.

It is known these magnitude features remain stable even when the character is shifted, which is the reason people used magnitude only. Phase features were proposed for Chinese character recognition in [1.16] for the first time with a detailed analysis, and were applied to design a tree classifier for 1000 Chinese characters.

Phase features are defined as

$$\theta_{ij} = \begin{cases} \text{Arctan}[Im(F_{ij})/Re(F_{ij})], & \text{if } Re(F_{ij}) > 0 \\ \text{Arctan}[Im(F_{ij})/Re(F_{ij})] + \pi, & \text{otherwise} \end{cases} \qquad (4.23)$$

The consideration of using phase features is twofold,

a) Information content of the magnitude features alone may not be enough for the recognition of thousands of Chinese characters.

b) Some character pairs have exactly the same magnitude features, but they differ in the phase features. They can not be distinguished by Fourier transforms without phase features. Let us take "甲" and "由" as an example, where pattern matrices are denoted by $A_{甲}$ and $A_{由}$. We have

$$A_{甲}(x,y) = A_{由}(8-x,8-y)$$

$$x=0,\ldots 7 \text{ and } y=0,\ldots,7,$$

$$\begin{aligned} F_{甲}(u,v) &= \sum_{x,y} e^{i(ux+vy)2\pi/8} A_{甲}(x,y) \\ &= [\sum_{x,y} e^{-i(u(8-x)+v(8-y))2\pi/8} \\ &\quad A_{由}(8-x,8-y)]\, e^{i(8u+8v).2\pi/8} \\ &= \overline{F}_{由}(u,v)\, e^{i(u+v).2\pi} \end{aligned}$$

$$u=0,\ldots,7 \text{ and } v=0,\ldots,7. \qquad (4.24)$$

The only difference is the phase part. Similar situation occurs between "人" and "入", "目" and "皿", and so on.

Some experiments were conducted using 20, 50, 100 and 200 characters respectively to test this conjecture. The magnitude features and phase features were computed and the results are shown in Table 4.4, where one can see that phase features really contain much information. Specifically when the character group is small, this information is relatively important. This was justified in the design of the tree classifiers [1.16], where magnitude features were mostly used at upper levels and phase features at lower levels.

However, in order to use phase features in Chinese character recognition, we have to overcome the following problems.

A) To make phase features stable under position shifting noise. This can be accomplished by finding the mean center $(m_x, m_y)$,

$$m_x = \sum_{i,j} i\, A_{ij}$$

$$m_y = \sum_{x,y} j\, A_{ij} \qquad (4.25)$$

and shifting the character such that its center is right at the matrix center. Because the center has a relatively stable position in the character, the shifting noise can be mostly eliminated.

B) To define the distance between two points along a phase feature axis. This is not trivial since the usual difference of two phases can not serve as a metric. As is

TABLE4.4 Average J measures of features

| No. of Categories | Magnitude Features | Phase Features |
|---|---|---|
| 20 | 176.7 | 171.5 |
| 50 | 145.5 | 47.2 |
| 100 | 127.8 | 48.6 |
| 200 | 123.2 | 38.7 |

TABLE4.5 Average Deviations of Phase Features (200 Categories) By 2 Methods

| No. | (1) | (2) | No. | (1) | (2) | No. | (1) | (2) |
|---|---|---|---|---|---|---|---|---|
| 1 | 0.1001 | 0.1001 | 11 | 0.1370 | 0.1374 | 21 | 0.0860 | 0.0885 |
| 2 | 0.0639 | 0.0639 | 12 | 0.1268 | 0.1268 | 22 | 0.0657 | 0.0657 |
| 3 | 0.0310 | 0.0310 | 13 | 0.1156 | 0.1180 | 23 | 0.0257 | 0.0257 |
| 4 | 0.1298 | 0.1306 | 14 | 0.0757 | 0.0757 | 24 | 0.0499 | 0.0499 |
| 5 | 0.1147 | 0.1153 | 15 | 0.0437 | 0.0437 | 25 | 0.0723 | 0.0723 |
| 6 | 0.1014 | 0.1044 | 16 | 0.0666 | 0.0666 | 26 | 0.1038 | 0.1038 |
| 7 | 0.0704 | 0.0704 | 17 | 0.0896 | 0.0919 | 27 | 0.1181 | 0.1187 |
| 8 | 0.0980 | 0.1000 | 18 | 0.1101 | 0.1105 | 28 | 0.0975 | 0.0975 |
| 9 | 0.0990 | 0.1053 | 19 | 0.1307 | 0.1322 | 29 | 0.0690 | 0.0705 |
| 10 | 0.1159 | 0.1160 | 20 | 0.1060 | 0.1060 | 30 | 0.0424 | 0.0424 |

Note: No. is the feature number

(1). Algorithm Mean - Deviation-1

(2). Algorithm Mean - Deviation-2

known, phase is a variable with periods of $2k\pi$, $k = 1, 2, \ldots$. The problem is which k we should use. In digital signal processing, this problem was solved by Oppenheim in 1965 ([4.15]) and later improved by others ([4.16]). Unfortunately their methods can not be used in our case for many reasons. Even the definition of sum and difference of two phase values have to be changed. The difference between two phase values is calculated along a unit circle instead of the Cartesian axis, as illustrated in Fig.6. One can see points x and y are near to each other along the circle but far apart from each other along the Cartesian axis. Under this consideration, the following definition is used :

$$d_{x,y} = \begin{cases} |x - y|, & \text{if } |x - y| \leq \pi \\ 2\pi - |x - y|, & \text{otherwise} \end{cases} \qquad (4.26)$$

where x, y are supposed to be in $(-\pi, \pi]$.

C) To define the mean value and deviation of a group of phase points, so that the statistical techniques are applicable to phase features.

DEFINITION 4.2. The mass center of mass points $m_1, m_2, \ldots, m_n$ at $x_1, x_2, \ldots, x_n$ along the unit circle is such a point x, that minimizes the value of

$$D = \sum_{i=1}^{n} m_i \, (d_{x,x_i})^2 \qquad (4.27)$$

and the corresponding value of D is the deviation.

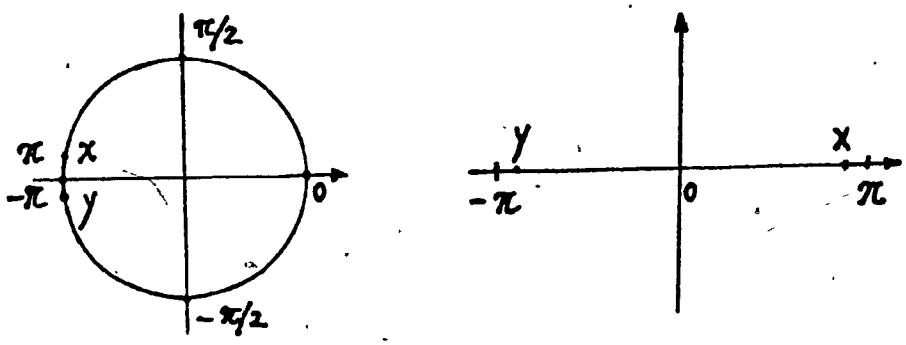
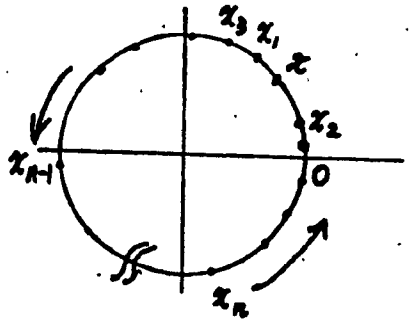



Fig. 4.6 Distance in the Phase Feature Space



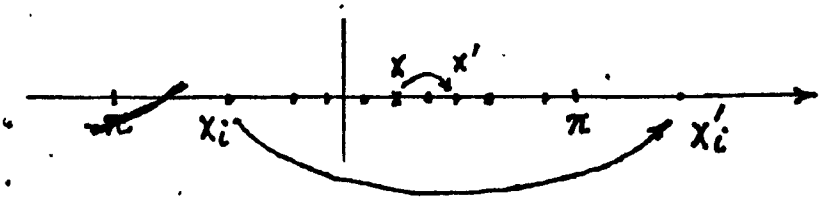




Fig. 4.7 Definition and Calculation of Mass Center

It is easy to show that this mass center exists. The problem is how to calculate it. Consider Fig.4.7, where x is the mass center of $x_1, x_2, \ldots, x_n$. If we shift the starting point of the circle to some point in the interval $(x_{i-1}, x_i]$, then x can be calculated using the traditional formula

$$x = \sum_i m_i x_i \Big/ \sum_i m_i \qquad (4.28)$$

We can calculate the value of x in (4.28) with the starting point at $x_1, x_2, \ldots, x_n$ alternatively, or equivalently shifting $x_i$ to $x_i' = x_i + 2\P$ consecutively for i=1,2,...,n-1. The mass center must be one of these values thus obtained, which minimizes D in eqn.(4.27).

Suppose the ith center candidate is $x^i$, and (see Fig.4.7)

$$x^{i+1} = x^i + 2\P\, m_i \Big/ \sum_j m_j \qquad (4.29)$$

which means the next candidate can be found by adding $2\P m_i / \sum_j m_j$ to $x^i$. After the point $x_i$ is shifted to $x_i' = x_i + 2\P$, the new value of D in eqn.(4.27) is

$$\begin{aligned} D^{i+1} &= \sum_j m_j (d_{x^{i+1}}, x_j)^2 \\ &= \sum_{j \neq i} m_j (d_{x^{i+1}}, x_j)^2 + m_i (d_{x^{i+1}}, x_i')^2 \\ &= \sum_j m_j (x^{i+1} - x_j)^2 \\ &\quad + m_i[(x^{i+1} - x_i')^2 - (x^{i+1} - x_i)^2] \end{aligned}$$

$$= \sum_j m_j (d_{x^i, x_j})^2 + (2\pi m_i)^2 / \sum_j m_j$$

$$+ [-2x^{i+1}x_i + 2x^{i+1}x_i + (x_i + 2\pi)^2 - x_i^2]m_i$$

$$= D^i + (2\pi m_i)^2 / \sum_j m_j - 2\pi m_i (2x^{i+1} - 2x_i - 2\pi) \quad (4.30)$$

Eqns.(4.29) and (4.30) lead us to the following algorithm :

ALGORITHM 4.4. Mean-Deviation-1.

Input : real numbers $x_1, x_2, \ldots, x_n$ in $(-\pi, \pi]$ in ascending order, positive real numbers $m_1, m_2, \ldots, m_n$.

Output : mass center x and deviation D as in Definition 4.2.

1) $S \leftarrow \sum_i m_i x_i \; / \; \sum_i m_i$

2) $T \leftarrow \sum_i m_i \, (d_{Sx_i})^2$

3) $D \leftarrow$ maxint

4) For $I \leftarrow 1$ to $n-1$ do

$S \leftarrow S + 2\pi \, m_i / \sum_j m_j$;

$T \leftarrow T + (2\pi/n)^2 \sum_j m_j$

$- 2\pi \, m_i (2S - 2x_i - 2\pi)$;

if $T < D$ then do

```
                x ← S;
                D ← T
            end
        end;
        5) Output the final values of x and D.
    end;
```

With the above algorithm, x can be calculated as in Definition 4.2, which minimizes the value of D in eqn.(4.27). D is actually the deviation of the points x's, if each $m_i/\Sigma_j m_j$ is understood as the corresponding probability.

D) To estimate the distribution of a character class in the feature space, given the sample patterns.

Normal distribution is often assumed in the numerical feature space. Although this is not always true, it usually works well. There are two reasons for people to use this assumption. One is the convenience in the computation of normal distribution. The other is that normal distribution is the least biased by the researcher, if the mean vector and the covariance matrix provide the only given information. This was proved in the J. Kampe de Fériet Theorem (1963, Chapter 16, [4.13]) :

The normal distribution maximizes entropy under given mathematical expectation and covariance matrix.

The phase features differ from other numerical features in that the value of each phase feature lies within the range of $(-\pi, \pi]$, which can not have normal distribution. Nevertheless, the way in which the J. Kampe de Fériet Theorem was proved can be similarly used to prove our following theorem :

THEOREM 4.3. Suppose $\Omega = (-\pi, \pi]$, and k is the set of Lebesgue measurable subsets of $\Omega$, the density $\rho(x)$ defines a probability measure $\nu$. If $\rho(x)$ maximizes the entropy

$$H(\rho) = -\int_{-\pi}^{\pi} \rho(x) \ln \rho(x)\, dx \qquad (4.31)$$

under the condition

$$Ef = \int_{-\pi}^{\pi} x\rho(x)\, dx = 0$$

$$Ef^2 = \int x^2\rho(x)\, dx = \sigma^2,$$

then $\rho(x)$ has the following form

$$\rho = (\beta/\pi)^{1/2} e^{-\beta x^2} / (1 - 2\Phi((2\beta)^{1/2}\pi)) \qquad (4.32)$$

where $\Phi$ is the normal distribution integration function

$$\Phi(x) = \int_x^{\infty} (2\pi)^{-1/2}\, e^{-t^2/2}\, dt \quad \text{for } x>0.$$

Proof.

We use the Lagrange multipliers $\alpha$ and $\beta$,

$$H(\rho) - \alpha - \beta\sigma^2 = \int_{-\infty}^{\infty} \rho(x) \ln (e^{-\alpha-\beta x^2}/\rho(x))\, dx$$

Using

$$\ln x \leq x-1,$$

we have

$$H(\rho) - \alpha - \beta x^2 \leq \int_{-\infty}^{\infty} e^{-\alpha-\beta x^2}\, dx - 1$$

The equality holds if and only if

$$\rho(x) = e^{-\alpha-\beta x^2}$$

Since

$$\int_{-\infty}^{\infty} \rho(x)\, dx = 1,$$

we have

$$e^{\alpha} = \int_{-\infty}^{\infty} e^{-\beta x^2}\, dx = [1 - 2\Phi((2\beta)^{1/2}\infty)]\,(\infty/\beta)^{1/2},$$

and

$$\rho(x) = (\beta/\infty)^{1/2} e^{-\beta x^2}/(1 - 2\Phi((2\beta)^{1/2}\infty)).$$

QED.

If $\sigma^2 << 1$, which is often the case for a good feature, then we have

$$\beta >> 1,\ \Phi((2\beta)^{1/2}\infty) \simeq 0,$$

and $\rho(\bar{x})$ is approximately normal in $[-\pi, \pi]$, as shown in Fig.4.8. This supports us to assume normal distribution when the phase feature satisfies

$$\sigma^2 << 1.$$

If all points are of equal mass, then Algorithm 4.1 can be simplified to become

ALGORITHM 4.2. Mean-Deviation-2.

1) M ← 0; D ← 0;

2) For i ← 1 to N do

M ← M + $x_i$;

D ← D + $x_i$ * $x_i$

end;

3) M ← M/n;

D ← D/n - M*M.

end;

When $\sigma^2 << 1$, this method gives very accurate results. An experiment was conducted with Algorithms 4.1 and 4.2 using 30 phase features extracted from 200 character classes, each with 10 samples. The average deviation values produced by these methods are shown in Table 4.5. There one

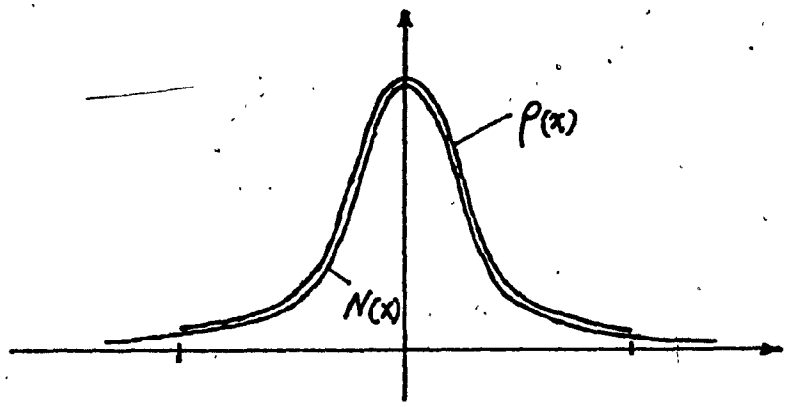



Fig. 4.8 Distribution ρ(x) and Normal Distribution N(x)

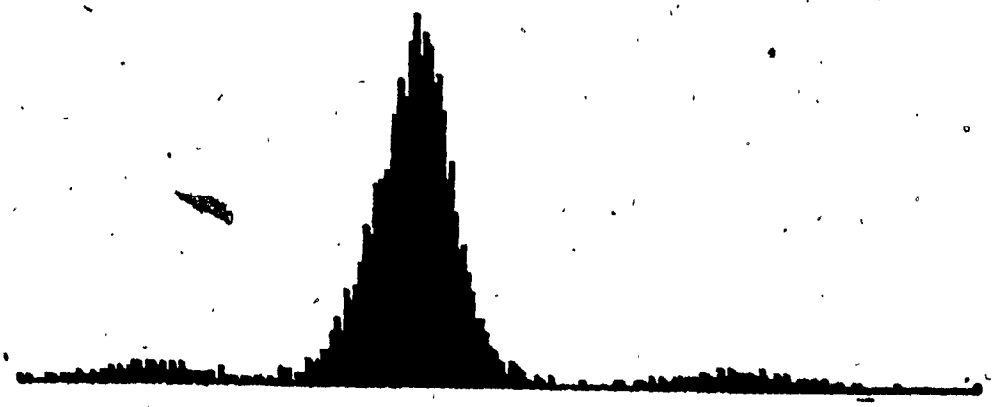


Fig. 4.9 A typical histogram of about 1000 characters.

can see when $D < 0.075$, both methods give the same results, when $D > 0.105$, Algorithm 4.2 gives slightly larger values than Algorithm 4.1.

For the application of phase features, see Chapter 6.

## 4.4. Information Content Feature Measure and Fisher's Criterion

Many feature measures have been studied for feature selection, e.g. see surveys [4.17, 4.18, 4.19, 4.25]. Of particular importance to multiple pattern class problems are those based on information content and scatter ratio, of which Fisher's criterion is a special case.

Information content is expressed by Shannon's entropy

$$H(\Omega/X) = E\{-\sum_{i=1}^{m} P(\omega_i|X) \log P(\omega_i|X)\} \qquad (4.33)$$

or mutual information

$$I(\Omega|X) = H(\Omega) - H(\Omega|X) \qquad (4.34)$$

where X is the feature or feature vector concerned, $\Omega$ the set of classes $\omega_1, \omega_2, \ldots, \omega_n$.

The scatter ratio, as an extension of Fisher's criterion, is defined as

$$J_2 = \log (\det (B + W)/\det W) \qquad (4.35)$$

where B is the average between-class covariance matrix

$$B = \sum_{i=1}^{m} P(\omega_i) (M_i - M_0) (M_i - M_0)^t \qquad (4.36)$$

and W the average within-class covariance matrix

$$W = \sum_{i=1}^{m} P(\omega_i E\{(X - M_i) (X - M_i)^t \mid \omega_i\}$$

$$= \sum_{i=1}^{m} P(\omega_i)\, \Sigma_i . \qquad (4.37)$$

$M_i$ is the mean vector of class $\omega_i$ and $M_0$ that of all samples. $P(\omega_i)$ is the a-priori probability of class $\omega_i$.

It is well known that

$$\ln x \simeq x - 1 \qquad (4.38)$$

or

$$\ln x \simeq x - 1 + (x - 1)^2/2 \qquad (4.39)$$

These approximations have been applied to information content measure, by which the average quadratic entropy measure $H_2$ and the average conditional cubic entropy measure $H_3$ have been proposed [4.17] :

$$H_2 = E\{\sum_{i=1}^{m} P(\omega_i|X)\, [1 - P(\omega_i|X)]\} \qquad (4.40)$$

$$H_3 = 1 - E\{\sum_{i=1}^{m} [P(\omega_i \mid X)]^3\} \qquad (4.41)$$

It is not intended to deduce any new measure in this section. Instead, (4.38) and (4.39) will be used to deduce some relationship between the two important measures given by eqns.(4.34) and (4.35) for one dimensional feature, under the following conditions [4.23].

Assume m is large, and the density $p(x|\omega_i)$ is normal

$$p(x|\omega_i) = (2\pi)^{-1/2}\, \sigma_i^{-1} \exp\{-(x - \mu_i)^2\, \sigma_i^{-2} / 2\}$$

$$i = 1, 2, \ldots, m.$$

By substituting them into (4.33), we can obtain the following result :

$$H(\Omega|x) = \sum_{i=1}^{m} P_i((\ln 2\pi)/2 + \ln \sigma_i) + \sum_{i=1}^{m} P_i - \sum_{i=1}^{m} P_i \ln P_i + H_1 \qquad (4.42)$$

where

$$H_1 = \int p(x) \ln p(x)\, dx \qquad (4.43)$$

In order to calculate $H_1$, we approximate the density function using the Hermite polynomial expansion. Under very general conditions, we have

$$p(x) = (2\pi)^{-1/2} \sigma^{-1} \exp[-(x - \mu)^2 \sigma^{-2}/2] \left\{ \sum_{i=1}^{\infty} c_j \psi_j(x - \mu) \right. \qquad (4.44)$$

where

$$\mu = \int x\, p(x)\, dx$$

$$\sigma^2 = \int x^2 p(x)\, dx - \mu^2$$

and each $\psi_j$ ($j = 1, 2, 3$) being a Hermite polynomial. Up to the 4th moment, we have [4.24, pp.188-189, and 1.2],

$$p(x) \simeq (2\pi)^{-1/2} \sigma^{-1} \exp\{-(x - \mu)^2 \sigma^{-2} / 2\} \{ 1 + m_3 \sigma^{-3} [((x - \mu)/\sigma)^3 - 3(x - \mu)/\sigma]$$

$$+ (m_4 \sigma^{-4} - 3)/4! \; [((x-\mu)/\sigma)^4 -$$

$$6((x - \mu) / \sigma)^2 + 3]]. \qquad (4.45)$$

Here

$$m_j = \int (x - \mu)^j \, p(x) \, dx \, , \; j = 1, 2, \ldots$$

using

$$p(x) = \sum_{i=1}^{m} p(x|\omega_i) P_i$$

we obtain

$$\mu = \sum_{i=1}^{m} P_i \mu_i$$

$$\sigma^2 = \sum_{i=1}^{m} P_i [\sigma_i^2 + (\mu_i - \mu)^2]$$

$$m_3 = \sum_{i=1}^{m} P_i [3\sigma_i^2 (\mu_i - \mu) + (\mu_i - \mu)^3]$$

$$m_4 = \sum_{i=1}^{m} P_i [3\sigma_i^4 + 6\sigma_i^2 (\mu_i - \mu) + (\mu_i - \mu)^4]$$

let us use the following abbreviations

$$\alpha = m_3 \sigma^{-3} / 3! \qquad (4.46)$$

$$\beta = (m_4 \sigma^{-4} - 3)/4! \qquad (4.47)$$

It is known that when $p(x)$ is normal,

$$\alpha = 0, \; \beta = 0.$$

If $p(x)$ is not far from normal, then $\alpha$ and $\beta$ are small and (4.43) becomes

$$H_1 = \int (2\pi)^{-1/2} \exp(-y^2 / 2)\, Q(y)\, dy \qquad (4.48)$$

where

$$\begin{aligned} Q(y) = {} & \beta^2 y^8/2 + (\alpha^2/2 - \beta^2/6 - \beta/2) y^6 \\ & + (21\beta^2 + 3\alpha^2 + 4\beta + 6\alpha - \beta(\ln 2\pi)/2 - \beta \ln \beta) y^4 \\ & + (-18\beta^2 + 9\alpha^2/2 - 15\beta/2 + 3\beta(\ln 2\pi) \\ & + 6\beta \ln \beta - 1/2) y^2 \\ & + (9\beta^2/2 + 3\beta^2 \ln \sigma - \ln \sigma) \\ & + \text{terms of } y \text{ to odd order} \\ & + o(\alpha^2 + \beta^2). \end{aligned}$$

Notice

$$\int \exp(-y^2/2)\, y^{2n}\, dy = (2\pi)^{1/2} (2n-1)!!$$

and

$$\int \exp(-y^2/2)\, y^{2n+1}\, dy = 0$$

(4.48) becomes

$$H_1 \simeq 3\alpha^2 + 12\beta^2 - (\ln 2\pi)/2 - \ln \sigma - 1/2 \qquad (4.49)$$

By eqns.(4.34), (4.42) and (4.49), we have

$$\begin{aligned} I &\simeq I_1(\Omega|x) \\ &\simeq \sum_{i=1}^{m} P_i \ln(\sigma/\sigma_i) - 3\alpha^2 - 12\beta^2 \qquad (4.50) \end{aligned}$$

If we use (4.38) instead of (4.39), the following approximation is obtained in exactly the same way (4.50) was deduced :

$$I \simeq I_2(\Omega|x)$$

$$\simeq \sum_{i=1}^{m} P_i \ln (\sigma/\sigma_i) - 6\alpha^2 - 24\beta^2. \qquad (4.51)$$

Just like

$$x - x^2/2 < \ln (1 + x) < x$$

we have approximately

$$I_2 < I < I_1. \qquad (4.52)$$

In one dimensional case, $J_2$ has the following form

$$J_2 = \ln (\sigma^2 / \sum_{i=1}^{m} P_i \sigma_i^2) \qquad (4.53)$$

Comparing eqns.(4.52) and (4.53), one finds

a) $J_2$ is similar to the major part of I with the difference that the average by $P_i$'s is inside logarithm in $J_2$, while outside in I.

b) The minor part of I, which $J_2$ does not have, is of order $o(\alpha^2+\beta^2)$, which is very small when the distribution is nearly normal.

c) I and $J_2$ do not differ much from each other. They both reflect the ratio of two factors, one being a

function of the between class variance, the other the within class variance. Whether using I or $J_2$, it is the same that the larger is the ratio, the better the feature's performance is considered to be.

To test the above assertions, some experiments were conducted.

1) Test of Normal Distribution assumption

P(x) was assumed normal or nearly normal in the deduction of eqn.(4.52), by which $\alpha^2$ and $\beta^2$ can be considered small. Tests were made for each of the Walsh transforms of the projection profiles of Chinese characters (see section 6.2), when the number of classes enlarged in the following manner

7, 27, 61, 108, 169, 243, 331, 432, 546 and 675.

The values of $\alpha^2$ and $\beta^2$ were found to be much smaller than 1 in these experiments. The average values of $\alpha^2$ and $\beta^2$ of the 64 features for each number of classes are shown in Table 4.6. These results reveal the following :

The more classes are considered, the smaller $\alpha^2$ and $\beta^2$ are. Even for small number of classes, $\alpha^2$ and $\beta^2$ are much smaller than 1. This fact confirms that the overall distribution, p(x) is approximately normal, and the deduction process to eqn.(4.52) is reliable for the large character set.

Tab. 4.6 Average Values of $\alpha^2$ and $\beta^2$

| No. of Classes | $\alpha^2$ | $\beta^2$ |
|---|---|---|
| 7 | 0.010368 | 0.001368 |
| 27 | 0.005305 | 0.001043 |
| 61 | 0.003009 | 0.000548 |
| 108 | 0.002243 | 0.000506 |
| 169 | 0.001975 | 0.000482 |
| 243 | 0.001844 | 0.000473 |
| 331 | 0.001889 | 0.000449 |
| 432 | 0.001291 | 0.000438 |
| 547 | 0.001060 | 0.000432 |
| 675 | 0.000970 | 0.000364 |

In addition, the histogram of the best feature of 1000 characters was plotted as shown in Fig.4.9. By examining this figure, one can see that when the number of classes is large, the distribution is practically normal.

2) Comparison of I and $J_2$.

To compare I with $J_2$, the following approximation is used

$$I \simeq \sum_{i=1}^{m} P_i \ln (\sigma/\sigma_i) - 4\alpha^2 - 16\beta^2 \qquad (4.54)$$

For 432 classes, the calculated values of measures I and $J_2$ are presented in Table 4.7. The features in this table have been arranged in descending order of measure I. For each feature, I and $J_2$ are also plotted in the feature-measure plane as shown in Fig.4.10, where we see : I and $J_2$ do not differ much, especially for several best features. They give nearly the same order of feaures.

For 243 classes, the distribution histogram of each feature is shown in Fig.4.11, where the features are arranged also in descending order of measure I. The histograms of the best and the worst features of 3155 characters are also shown in Fig.4.12. From these histograms, we can see that the approximation (4.54) of I really reflects separability, or information of each feature.

3) Recognition test

Table4.7 Four measures of 64 features - 432 classes in descending order of $I_0$.

| F | I0 | J0 | J1 | J2 | F | I0 | J0 | J1 | J2 |
|---|---|---|---|---|---|---|---|---|---|
| W 2 | 2.990 | 2.994 | 2.946 | 2.901 | W27 | 2.181 | 2.181 | 2.073 | 1.949 |
| W 1 | 2.983 | 2.987 | 2.939 | 2.895 | W32 | 2.180 | 2.195 | 2.088 | 1.980 |
| W10 | 2.907 | 2.910 | 2.855 | 2.802 | W40 | 2.164 | 2.191 | 2.089 | 1.984 |
| W 6 | 2.880 | 2.894 | 2.836 | 2.785 | W33 | 2.163 | 2.165 | 2.069 | 1.966 |
| W 5 | 2.820 | 2.824 | 2.767 | 2.715 | W29 | 2.162 | 2.167 | 2.074 | 1.975 |
| W14 | 2.734 | 2.741 | 2.683 | 2.624 | W54 | 2.133 | 2.170 | 2.087 | 2.004 |
| W22 | 2.669 | 2.674 | 2.590 | 2.508 | W36 | 2.130 | 2.227 | 2.124 | 2.017 |
| W18 | 2.629 | 2.649 | 2.570 | 2.493 | W46 | 2.119 | 2.192 | 2.104 | 2.014 |
| W26 | 2.600 | 2.651 | 2.563 | 2.466 | W49 | 2.112 | 2.115 | 2.028 | 1.945 |
| W 9 | 2.526 | 2.549 | 2.497 | 2.449 | W44 | 2.103 | 2.121 | 2.029 | 1.927 |
| W12 | 2.513 | 2.514 | 2.425 | 2.330 | W37 | 2.098 | 2.107 | 2.016 | 1.912 |
| W17 | 2.506 | 2.512 | 2.430 | 2.347 | W48 | 2.076 | 2.084 | 1.992 | 1.897 |
| W16 | 2.485 | 2.485 | 2.399 | 2.311 | W31 | 2.069 | 2.069 | 1.961 | 1.846 |
| W30 | 2.422 | 2.472 | 2.369 | 2.254 | W39 | 2.065 | 2.065 | 1.970 | 1.869 |
| W25 | 2.373 | 2.379 | 2.293 | 2.203 | W50 | 2.053 | 2.101 | 2.025 | 1.947 |
| W 7 | 2.362 | 2.384 | 2.310 | 2.229 | W58 | 2.020 | 2.095 | 2.013 | 1.931 |
| W13 | 2.360 | 2.364 | 2.300 | 2.238 | W43 | 2.013 | 2.016 | 1.915 | 1.806 |
| W 4 | 2.330 | 2.330 | 2.245 | 2.160 | W41 | 1.999 | 2.001 | 1.887 | 1.764 |
| W15 | 2.322 | 2.350 | 2.275 | 2.193 | W45 | 1.986 | 1.991 | 1.896 | 1.786 |
| W21 | 2.315 | 2.325 | 2.246 | 2.161 | W47 | 1.924 | 1.962 | 1.870 | 1.771 |
| W20 | 2.314 | 2.317 | 2.221 | 2.119 | W52 | 1.892 | 1.903 | 1.820 | 1.738 |
| W 8 | 2.312 | 2.343 | 2.264 | 2.190 | W57 | 1.878 | 1.900 | 1.806 | 1.711 |
| W24 | 2.299 | 2.307 | 2.208 | 2.101 | W51 | 1.864 | 1.900 | 1.800 | 1.694 |
| W38 | 2.296 | 2.299 | 2.206 | 2.107 | W53 | 1.863 | 1.877 | 1.791 | 1.705 |
| W34 | 2.294 | 2.302 | 2.194 | 2.080 | W62 | 1.794 | 1.813 | 1.732 | 1.655 |
| W11 | 2.272 | 2.288 | 2.216 | 2.128 | W56 | 1.773 | 1.795 | 1.707 | 1.618 |
| W28 | 2.259 | 2.275 | 2.161 | 2.050 | W59 | 1.685 | 1.686 | 1.600 | 1.509 |
| W19 | 2.255 | 2.270 | 2.187 | 2.096 | W55 | 1.646 | 1.737 | 1.640 | 1.537 |
| W23 | 2.220 | 2.252 | 2.163 | 2.062 | W61 | 1.562 | 1.576 | 1.502 | 1.421 |
| W 3 | 2.219 | 2.239 | 2.166 | 2.069 | W60 | 1.472 | 1.591 | 1.517 | 1.445 |
| W42 | 2.195 | 2.273 | 2.181 | 2.088 | W63 | 1.292 | 1.295 | 1.215 | 1.136 |
| W35 | 2.183 | 2.184 | 2.088 | 1.987 | W64 | 1.125 | 1.154 | 1.085 | 1.018 |

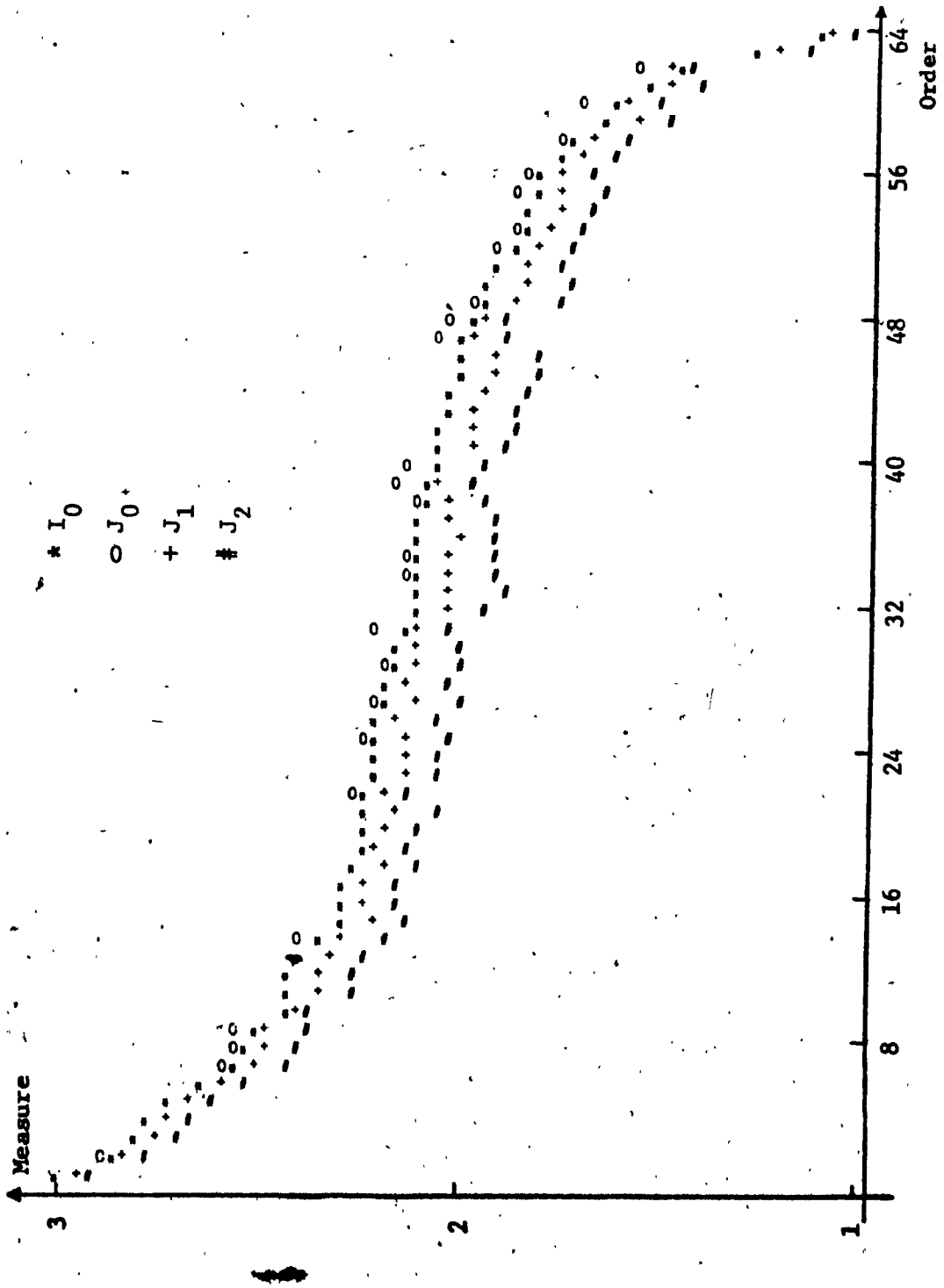



Fig. 4.10 Feature ordering, 432 classes.

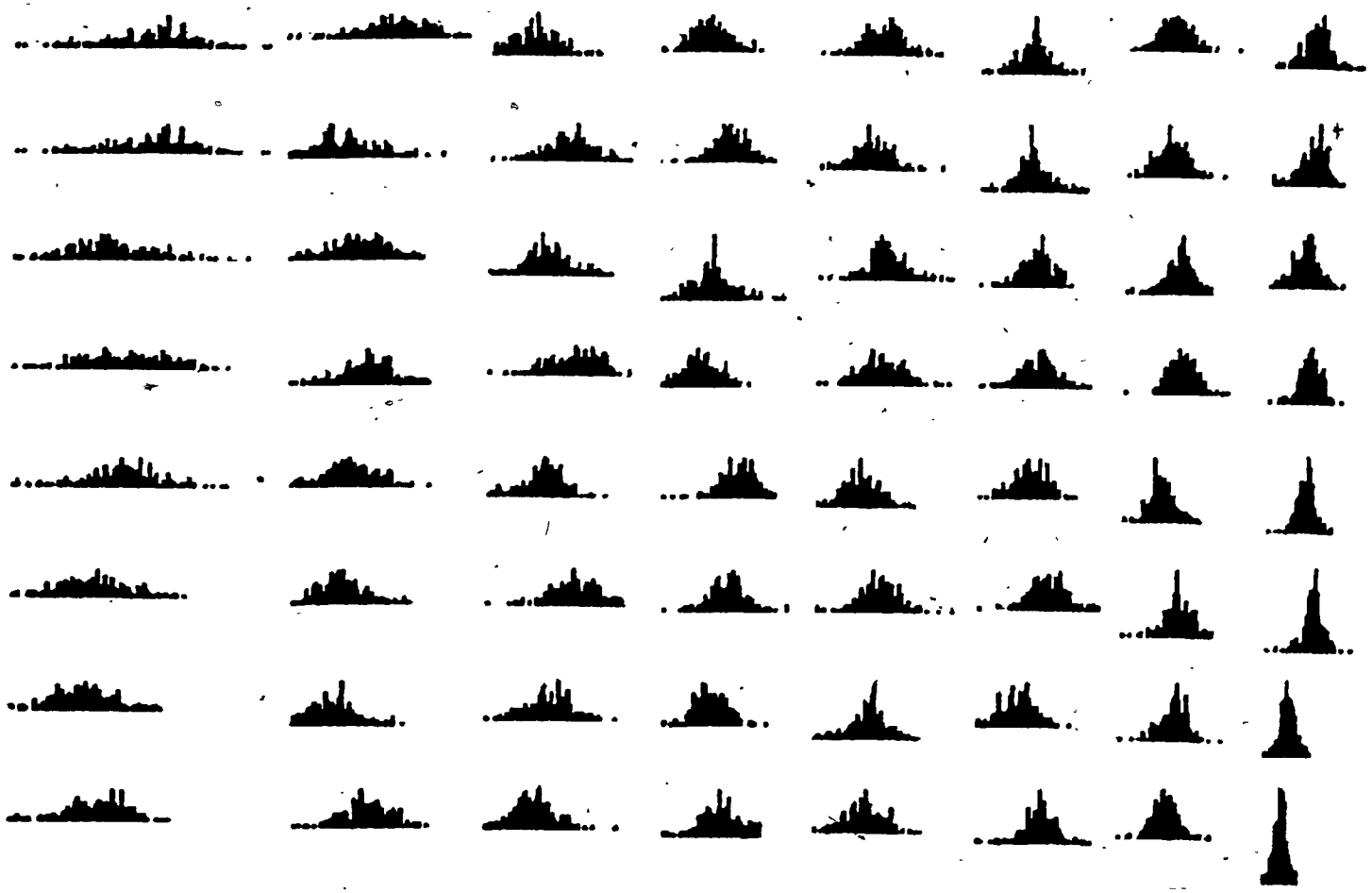


Fig. 4.11 Histograms of ordered features, 243 classes.

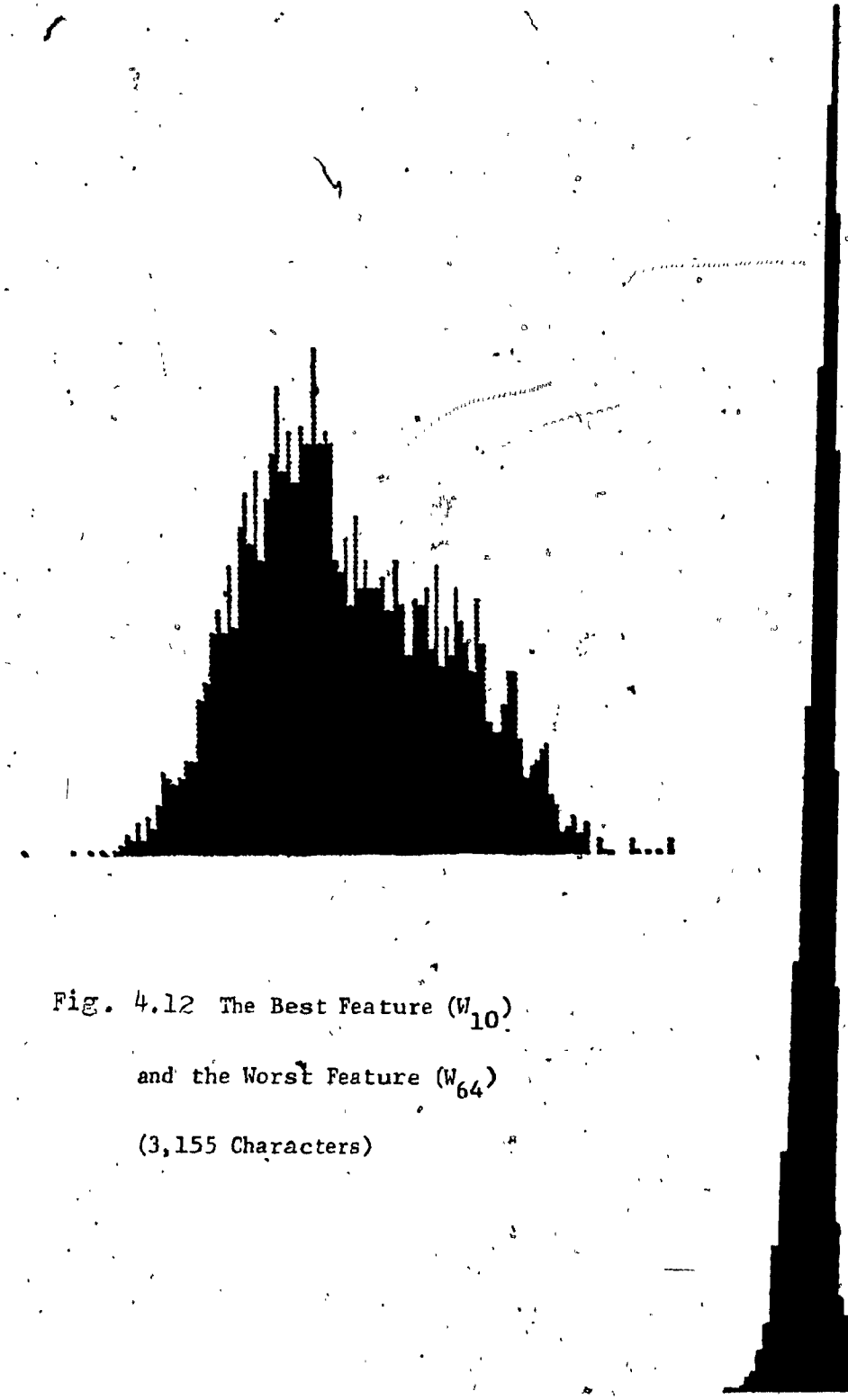


Fig. 4.12 The Best Feature ($W_{10}$)
and the Worst Feature ($W_{64}$)
(3,155 Characters)

In the design of the decision tree with one feature used in each internal node, the feature selection is simply to find the best feature. Both (4.53) for $J_2$ and (4.54) for I are applicable. Two such trees for 206 classes each using $J_2$ or I were designed and the recognition rates were tested using independent testing samples. The results are presented in Table 4.8, where $J_2$ and I show nearly the same performance in overlap and recognition rate.

At the first glance, the Shannon's information content measure given in (4.33) and the scatter ratio or Fisher's criterion in (4.35) look so different. But in a multi-class problem like the Chinese character set, the above experiments show that they behave in a very similar manner. Under a reasonable assumption that each class has a normal distribution and the overall distribution is nearly normal, formula (4.52) for I was deduced, which looks very similar to $J_2$ in (4.53). All these experiments and deduction reveal that $J_2$ and I count essentially the discriminating power of features in a very similar way when the number of classes is very large.

In the main line of tree analysis in this thesis, entropy reduction is emphasized, which has a close relation with information content measure. While this measure is complex to compute, yet when we design a large tree, feature selection need be done hundreds or even thousands of times, we need a simpler one. $J_2$ serves this purpose very well,

Table 4.8. Decision Trees Using $I_0$ and $J_2$ (206 Classes)

| description | $I_0$ | $J_2$ |
|---|---|---|
| internals | 160 | 150 |
| terminals | 290 | 287 |
| overlaps | 84 | 75 |
| recognition rates | 98.5% | 97.9% |

and still keeps the same style as information content measure. Due to this reason, in Chapter 6, most experiments are associated with feature selection by $J_2$.

## 4.5. References


[1]. T. Pavlidis, Structural Pattern Recognition, Springer - Verlag, Berlin, New York, 1977.

[2]. K. Nakata, Y. Nakano and Y. Uchikura, "Recognition of Chinese characters," Proceedings of the Conference on Machine Perception of Pattern and Picture, Teddington, pp.45-52, April, 1972.

[3]. Y. Nakano, K. Nakata, Y. Uchikura and A. Nakajima, "Improvement of Chinese character recognition using projection profiles," Proc. of the 1st Int. Joint Conf. on Pattern Recognition, pp.172-178, 1973.

[4]. K. L. Chai, W. Y. Young and P. F. Chen, "The recognition of Chinese characters using black pel density for preliminary classification and amplitude spectrum of projection profiles," Proc. Int. Computer Conf. 8-2.2 to 8.2.13, 1980.

[5]. Q. R. Wang and C. Y. Suen, "SVD and profiles of Chinese characters," Technical Report, Dept, of Computer Science, Concordia University, Dec. 1981.

[6]. Q. Y. Shi, Image restoration and Digital filter, Peking University Publishing Hall, in Chinese, June, 1979.

[7]. N. Otsu, "Picture representation by meginal eigen vectors," Bul. Elec. Lab., vol.39, No.12, 1975.

[8]. A. Albert, Regression and the Moore-Penrose Pseudoinverse, Academic Press, New York, 1972.

[9]. T. S. Huang (ed.), Picture Processing and Digital Filtering, Springer-verlag, Berlin, Heidelberg, New York, 1975.

[10]. H. C. Andrews and B. R. Hunt, Digital Image Reconstruction, Prentice-Hall Inc., Englewood Cliffts, New Jersey, 1977.

[11]. C. Y. Suen, "Feature extraction in automatic recognition of handprinted characters," Signal Processing, vol. 4, Nos.2,3, 193-207, April 1982.

[12]. H. C. Andrews, "Multidimensional rotation in feature selection," IEEE Trans. on Computer, vol. 20, pp. 1045-1051, Sept. 1971.

[13]. Guiasu Silviu, Information Theory with Applications, McGraw-Hill, New York, 1977.

[14]. S. T. Bow and W. T. Kim, "Some consideration on the machine recognition of Chinese ideographs," Proc. Int. Conf. on the Chinese Computer Society, 333-343, Sept. 1982.

[15]. A. V. Oppenheim and R. W. Schafer, Digital Signal Processing, Englewood Cliffs, N. J.: Prentice-Hall, 1975.

[16]. J. M. Tribolet, "A new phase unwrapping algorithm," IEEE Trans. Acoust. Speech Signal Processing., ASSP-25, 170-177, 1977.

[17]. C. H. Chen, "On the use of distance and information measures in pattern recognition and applications," in Pattern Recognition : Theory and Applications, K. S. Fu (ed.), pp. 45-60, Noordhoff - Leyden, 1977.

[18]. J. Kittler, "Feature selection methods based on the Karhunen-Loëve expansion," ibid., pp.61-74.

[19]. H. F. Ryan, "The information content measure as performance criterion of feature selection," Proc. of the 7th IEEE Synposium on Adaptive Processes, pp. 2-c-1-2-c-11, 1968.

[20]. E. Yamamoto, N. Fujii, T. Fujita, C. Ito and J. Tanahashi, "Handwritten Kanji character recognition using the features extracted from multiple standpoints," Proc. IEEE Conf. on Image Processing and Pattern Recognition pp. 131-136, Dallas, Aug. 1981.

[21]. M. T. Cheng, S. C. Shen and M. J. Chow, "Finite Walsh Transform in $R^n$ and Application to Picture Bandwidth compression," Journal of Science, Peking University, pp.25-50, 1979.

[22]. Q. R. Wang, Y. X. Gu and C. Y. Suen, "A preliminary study on computer recognition of Chinese characters

printed in different fonts," Proc. Int. Conf. of the Chinese Language Computer Society, 344-351, Sept. 1982.

[23]. Q. R. Wang, Y. X. Gu and C. Y. Suen, "Fourth moment measure and its application in the selection of features from multi-class pattern," Technical Report, Dept of Computer Science, Concordia University, sept. 1981.

[24]. A. Papoulis, Probability, Random Variables and Stochastic Processes, McGraw-Hill, New York, (Chapters 2, 3), 1965.

[25]. C. H. Chen, "On a class of computationally efficient feature selection criteria", Pattern Recognition, Vol. 7, No. 1, 1975.

# CHAPTER FIVE

# A NEW TREE MODEL WITH IMPROVED SEARCH ALGORITHM

## INTRODUCTION

Is has been pointed out in Chapter 2 that the tree classifier with conventional search suffers from error accumulation. In this chapter new search algorithms are proposed which employ fuzzy logic as search heuristic and terminal similarity for decision making. An extension of the similarity measure is introduced to enhance the search algorithm. Based on the above consideration, a new decision tree model is proposed, which solves the error accumulation problem.

## 5.1. Straight Forward Search and Error Accumulation

Let us consider a general tree which has been mapped into a binary tree in the mempry (Section 1.3). Suppose l features are used in each internal node and a distance or dissimilarity measure is used as the descriminant. To illustrate this, an example is given in Fig.5.1, where l = 4 and the Euclidean distance is used.

Using the data structure introduced in Section 1.3, the conventional search method can be summarized as follows :

ALGORITHM 5.1. Straight-Forward-Search.

Input : the unknown pattern described by its feature vector $F_{[1..n]}$.

The tree is supposed to be in the memory.

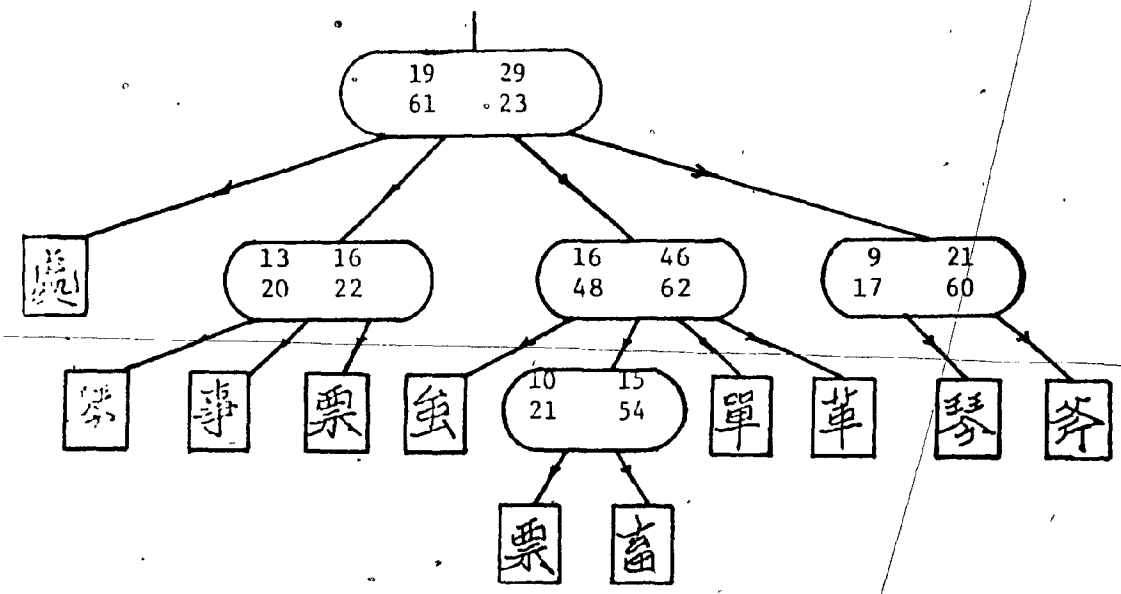
Output : the character class i.d., to which the unknown is assumed.

1) Take the root of the tree as the current internal node.

2) Get feature number $i_1, i_2, \ldots, i_l$ from FN in the current node.

3) Let $X = (F_{i_1}, F_{i_2}, \ldots, F_{i_l})^t$.

4) Find the number of child nodes, k, and the k children



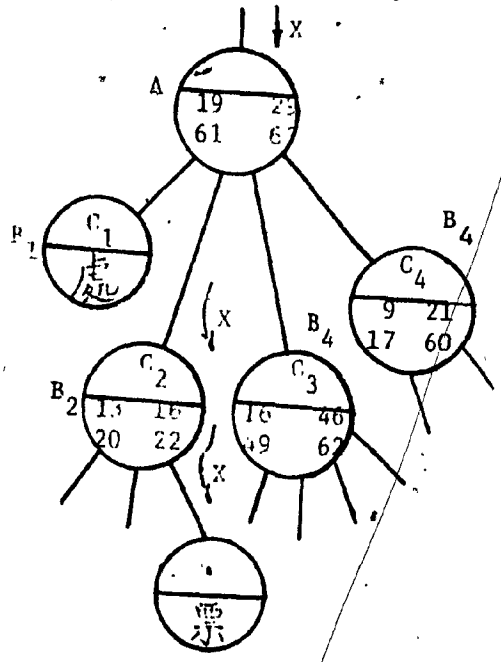




FIG.5.1 Straight Forward Search

$B_1, B_2, \ldots, B_k$ (by pointer manipulation). Find the corresponding $C_1, C_2, \ldots, C_k$ (the FEA field in these child nodes) respectively located in the 1-dimensional subspace.

5) Compute the normalized distance between $\tilde{X}$ and $C_j$ :

$$d_j = (X - C_j)^t \, \mathrm{diag}(S_{i_1}^{-2}, \ldots, S_{i_l}^{-2}) \, (X - C_j),$$

$$j = 1, 2, \ldots, k \qquad (5.1)$$

where $S_{i_j}$ is the standard deviation of the $i_j$th feature.

6) Select a child node with the smallest value of $d_j$.

7) If this node is a terminal, then the unknown pattern is recognized. Output the i.d. indicated in the terminal node, terminate the process.

Otherwise this node is taken as the current internal node and the above procedure is repeated from step 2.

END

This algorithm was used in the simulation experiments described in [1.14, 1.15]. There are two shortcomings in this algorithm :

1) No rejection is made. Any misrecognition introduces an error. It is well known that error is much worse than rejection, since the latter may be remedied by post-processing, while nothing can help if an error occurs in a real process.

2) In the search process, the error committed at any internal node can not be recovered and the total error rate is the accumulation of error rates at all levels of the tree. As shown in Chapter 2, the error rate of the decision tree is in the order of $H \simeq \log n$, based on the assumption the straight forward search is actually used.

For Chinese character recognition, the tree has many levels and error accumulation may not be permitted. Two sophisticated search algorithms will be presented in the following sections to solve this problem.

## 5.2. Fuzzy Logic Search

Fuzzy decision tree search was first proposed and analyzed in [5.2], where the BBB algorithm (Branch-Bound-Back, also see [5.3]) was given with satisfactory analysis and experiment. We introduce a new concept which is similar to those proposed in [5.2]. But in Chinese character recognition, both straight forward search and fuzzy logic search are necessary for the sake of speed and recognition rate. The tree is the same as before. We rather call it a decision tree "with fuzzy logic search". The search algorithm also belongs to the branch-bound family, with the width first strategy.

Fuzzy set theory was first introduced by Zadeh in 1965 [5.4] and developed in these two decades. Rather than "probability", fuzziness is considered a concept of "possibility". The mathematical definitions of these two uncertainties are different and the latter is not a proper description of things like "error probability". Instead, fuzziness is used as a heuristic evaluation in the tree search and it has been found to work well, as will be seen in the next Chapter.

Consider the unknown pattern X shown in Fig.5.1, which is going to reach one of the child nodes. The only pieces of information available are the values of $d_j$'s in

eqn.(5.1). We have to find out to what degree X likely belongs to $B_j$, using the information given by the $d_j$'s. Suppose we have

$$d_{j_1} < d_{j_2} < \ldots < d_{j_k} .$$

Let Y be the set of all unknown patterns. We can define the following fuzzy subsets

$$F_j = \int_Y \mu_j(X) / X, \; j = 1, 2, \ldots, k \qquad (5.2)$$

where the membership functions are given below :

$$\mu_{j_1} = 1 \wedge \epsilon \cdot d_{j_2} / (d_{j_1} + d_{j_2}),$$

$$\mu_j = \epsilon \cdot d_{j_1} / (d_{j_1} + d_j),$$

$$j = 1, 2, \ldots, k \text{ but } j \neq j_1 \qquad (5.3)$$

where $\epsilon$ is a constant selected as

$$0 < \epsilon < 1,$$

When $k = 2$, the relation between $\mu_1 - \mu_2$ and $d_1 - d_2$ is drawn in Fig.5.2, where we can see that the bigger $\mu_1$ is, the nearer is X to $C_1$ and the more likely X belongs to $B_1$. In this way the "possibility" of "X belonging to group j" is described by fuzzy subset $F_j$.

Let us consider Fig.5.3, where the unknown has reached $F_2$ and $G_1$, $G_2$, $G_3$ are the membership functions of the child nodes. $G_1$, $G_2$ and $G_3$ can be defined in the same manner as

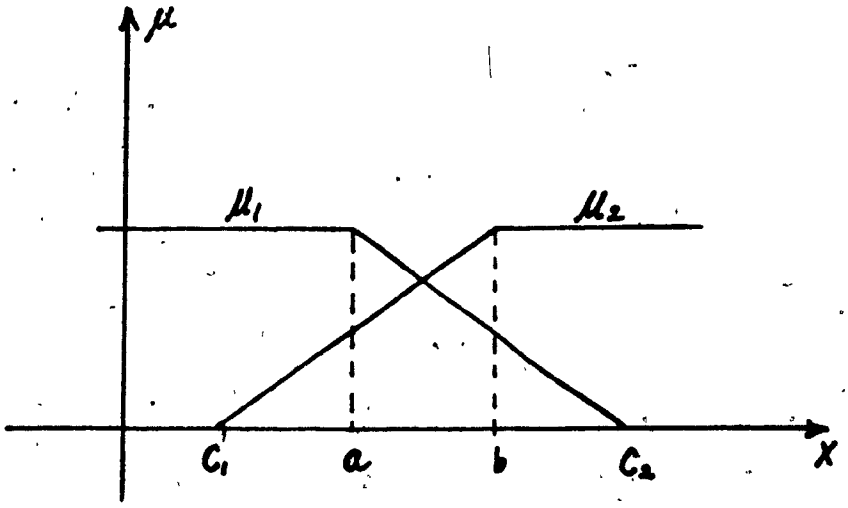



FIG. 5.2 Membership Function μ(x)

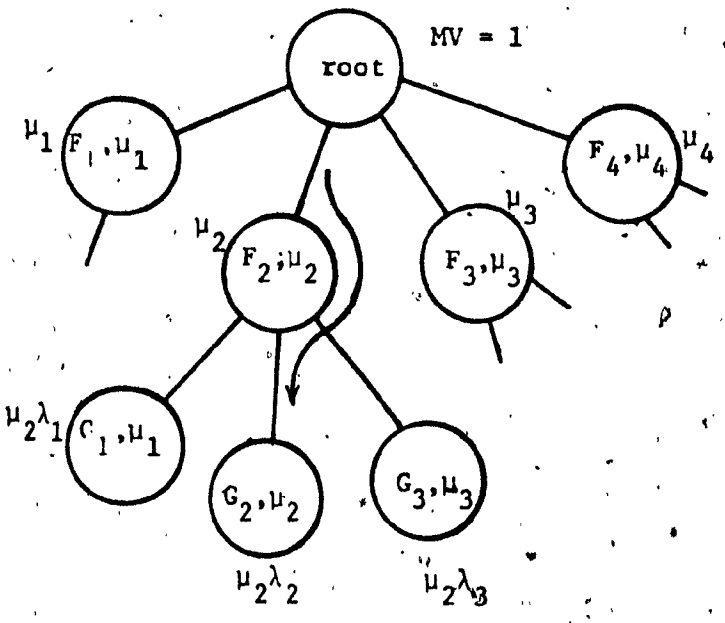



FIG. 5.3 Membership Value MV by Multiplying Rule

in eqn.(5.3) by means of the distance between X and each new centroid. Each $\mu_j$ represents the degree of likelihood that X belongs to $F_j$, while each $\lambda_i$ represents the degree of likelihood that X belongs to $G_i$ under the condition that X has been known to belong to $F_2$. Then the membership function for an unknown X to belong to $G_i$ is

$$MV_{G_i} = \mu_2 \lambda_1, \quad I = 1, 2, 3 \qquad (5.4)$$

In this way we are using the multiplying rule of conjunctions [5.5] :

$$F \langle and \rangle G \triangleq \int_X \mu_F(X) \mu_G(X) / X \qquad (5.5)$$

There is another definition of conjunction, called minimum rule,

$$F \langle and \rangle G \triangleq \int_X \mu_F(X) \wedge \mu_G(X) / X \qquad (5.6)$$

The reason that we did not use this minimum rule is that it does not record enough information of the "past" in the search. For example, if

$$\mu_F = 1/2, \mu_G = 1/2,$$

then

$$\mu_{F \langle and \rangle G} \triangleq 1/2 \text{ by the minimum rule}$$

and

$$\mu_{F \langle and \rangle G} = 1/4 \text{ by the multiplying rule.}$$

Obviously the former contradicts our intuition. Some experiments were conducted and their results supported the multiplying rule as the psychological process of conjunction [5.6], which coincides with our selection.

Now we are ready to state our search algorithm. In the decision tree, in addition to the data structure in Section 1.3, each node contains an MV field to store the membership value of the unknown. A link list of potential nodes, LP, is kept. The input is an unknown pattern. The output includes recognition or rejection. The latter was not included in the straight forward search. In the new algorithm, the straight forward search is used first. At the terminal which has been reached, if the distance between the unknown pattern and the center of the character class indicated in the terminal is small enough, then decision is made, otherwise the fuzzy logic search is invoked. This algorithm was developed in [1.16], which is presented below.

ALGORITHM 5.2. Fuzzy-Logic-Search-1.

1) Find out a terminal node by the straight forward search.

2) If the distance (or dissimilarity) between the unknown pattern and the known class is smaller than an established threshold, then output the recognition result and terminate the process, otherwise go to step 3.

3) Initialize the list LP with only the root node in it. The value of MV in the root node is set to 1.

4) If LP is empty then output rejection and terminate the process, otherwise go to step 5.

5) Take off the first node from LP. If it is a terminal node then output recognition result and terminate the process, otherwise go to step 6.

6) For each of its child nodes, calculate the $MV_i$ value

$$MV_i = MV^{\mu_i}$$

where $\mu_i$ is as given in eqn.(5.3).

7) put all child nodes with $MV_i \geq \delta$ (a given lower bound) in the LP list and sort LP in descending order of MV values. Go to step 4.

END

Step 1 is the straight forward search, as in algorithm 5.1. It takes O(log n) time. Since the recognition rate of a decision tree is not low (98% or even higher based on our experience), most unknown patterns are already recognized properly in this step. If the terminal node found in this step is a wrong one, in most cases, the dissimilarity will be large and the fuzzy logic search is invoked to begin at step 2. Steps 4--7 form a loop. The time taken depends on

how many new child nodes are put in the LP list each time step 7 is executed. Because the tree is designed using ISOETRP, overlap is controlled and the probability for a sample to be in the interval (a, b) in Fig.5.2 is very small. Very often only one child node has a high value of MV, or sometimes two child nodes have $MV \geq \delta$ and the others $< \delta$. Thus the fuzzy logic search actually does not access many terminals, much less than n (n is the number of terminal nodes accessed in the worst case, when $\delta = 0$). In this way, this algorithm is very fast on the average, although it could take more time than the straight forward search for a small percentage of unknowns.

It is important to analyze the admissibility of a search algorithm, which can be defined as follows [5.1] for our case :

DEFINITION 5.1. The algorithm is said to be admissible if it always finds the terminal node with the largest MV value, if $MV \geq \delta$.

It is easy to show that Algorithm 5.1 is admissible, which is written in the following Theorem :

THEOREM 5.1. Algorithm 5.2 is admissible in the sense of Definition 5.1.

Proof.

Assume T is the terminal node with the largest value of MV among all the terminals of the tree. If

$$MV_T \geq \delta$$

then all ascendants of T, including the root node, have their MV values larger than or equal to $\delta$. Assuming they are

$$T_0, T_1, \ldots, T_k$$

with $T_0$ being the root node and $T_i$ the parent node of $T_{i+1}$, $i = 0, 1, \ldots, k-1$, and $T_k = T$.

When the search starts at step 3, $T_0$ is in LP. Obviously, if any $T_i$ is taken off from LP at step 5, then $T_{i+1}$ must be put into LP at step 7 whenever $i < k$, since their MV $\geq \delta$.

In this way the algorithm can not terminate before $T_k$ is reached. If it terminates at step 5, $T_k = T$ is found since it has the largest MV value among terminals and it is in front of any other terminals by sorting at step 7. QED.

If we take $\delta = 0$, then Theorem 5.1 says that, the algorithm always finds the terminals with the largest MV value among all terminals. But if this MV value is too small, it is not worthwhile to make decision. It is better to set $\delta$ above 0 in a proper manner and rejection is made

when there is no terminal node with $MV \geq \delta$. Accordingly the number of internal nodes put in LP can be controlled by adjusting $\delta$. This is useful in handling searching time. When $\delta = 0$, it takes the longest time, which is often not worth the trouble.

## 5.3. Improvement of the Fuzzy Logic Search

There is still room in the above algorithm for further improvement. Firstly, the membership value defined in (5.3) is subjective. It is, in fact, used as heuristic evaluation of the nodes. Whether it works well or not depends on how well it reflects the likelihood of the unknown belonging to the corresponding subset.

If each class occupies an ellipsoidal region as assumed in eqn.(2.10),

$$\Omega_i = \{\underline{X} \mid (\underline{X} - \underline{C}_i)^t \Sigma_i^{-1} (\underline{X} - \underline{C}_i) < \alpha\},$$

$$i=1, 2, \ldots, k$$

with a probability approaching one, then the center of the possible class to which the unknown pattern $\underline{X}$ belongs lies in the region

$$\Omega_X = \{C \mid (C - X)^t \Sigma_i^{-1} (\underline{C} - \underline{X}) < \lambda\alpha\} \qquad (5.7)$$

with a probability approaching one, where $\lambda$ is a constant for adjustment. This is depicted in Fig.5.4, where $C_1$ and $C_2$ are group centers. It is seen that, if

$$d_2^2 - d_1^2 < \lambda\alpha,$$

where $d_1$ and $d_1$ are the distances between X and $C_1$ and $C_2$ respectively, then there is the possibility that X belongs

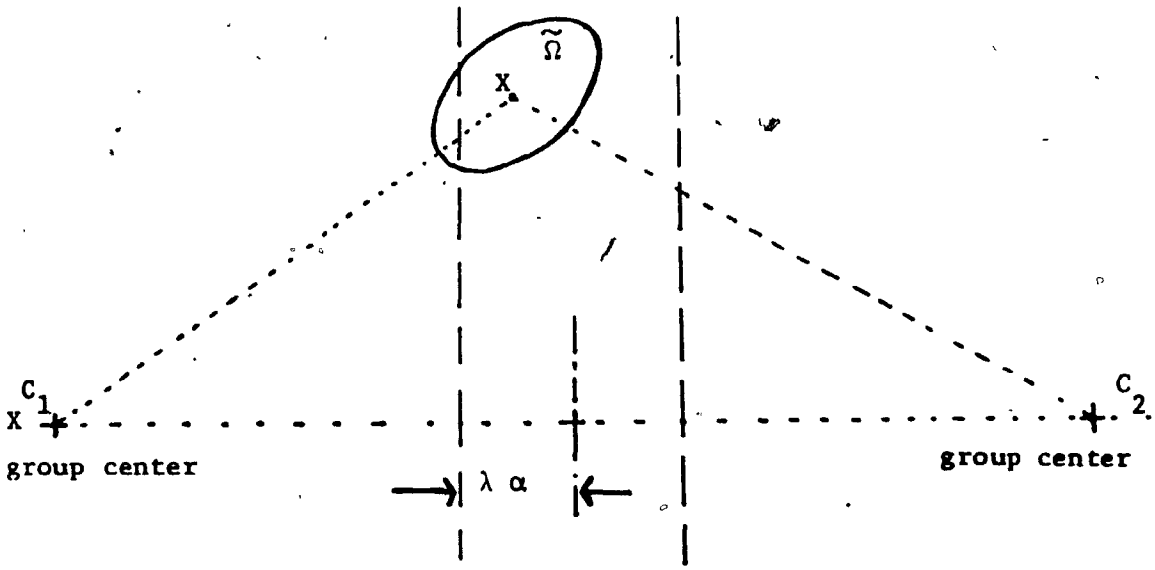
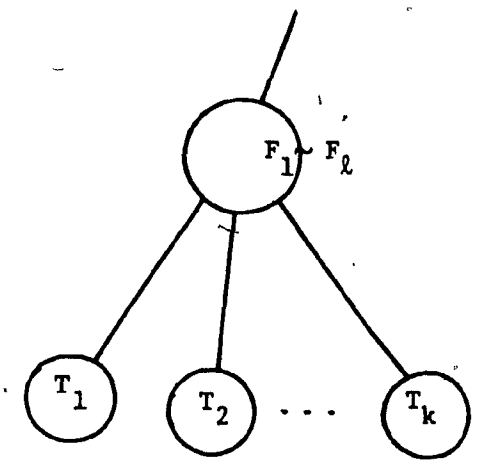
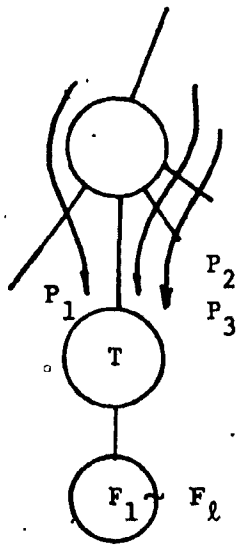



Fig. 5.4 Region of Possible Class Centers




a. Considering adjacent terminals




b. Considering Possibly Coming Patterns

Fig. 5.5 Similarity Measures in a Conventional Tree and a Globally Trained Tree

to a class in group $C_2$, although it is nearer to $C_1$ and most likely it belongs to group $C_1$. Based on this consideration, eqn.(5.3) is modified as

$$\mu_{j_1} = 1 \wedge [\ (d_{j_2}^2 - d_{j_2}^2)/(\lambda\alpha) \vee 0\ ] \qquad (5.8)$$

$$\mu_j = 0 \vee [0.5-(d_j^2-d_{j_1}^2)/(\lambda\alpha)]$$

$$j = 1, 2, \ldots, k \text{ but } j \neq j_1, \qquad (5.9)$$

where $d_1, d_2, \ldots, d_k$ are computed by eqn.(5.1) and ordered as

$$d_{j_1} \leq d_{j_2} \leq \cdots \leq d_{j_k}.$$

Equations (5.8) and (5.9) were used as a heuristic in a simulation experiment in Chapter 6, which showed better performance than eqn.(5.3).

The second improvement is the decision making at step 5. In algorithm 5.2, only the terminal node with the largest MV value is examined. But if there are several terminals, which have approximately equal MV values, it is not worthwhile making a decision in this way. Even in the case a certain terminal may have an MV value much larger than all the others, it is actually not a sophisticated consideration to make decision because MV value is nothing but a heuristic evaluation. No relation between MV and error rate is guaranteed. We should distinguish between the functions of MV value and of similarity (or dissimilarity, or

distance). The latter may have some relation with the error rate in a concrete probabilistic model, while the former is used only to guide the search. A possible improvement is to introduce a similarity measure in decision making and examine all terminals with MV values above $\delta$. For this purpose, the sort procedure in step 7 in Algorithm 5.2 is no longer necessary.

The third improvement is the introduction of extended similarity.

Recall step 2 of Algorithm 5.2, where dissimilarity was used at terminals. This dissimilarity is specially designed for the terminals under the same parent node. It is only good in distinguishing the "terminal siblings of the same parent". But in the fuzzy logic search, the pattern reaching one of these terminals with a membership value MV$\geq^{\delta}$ (Fig.5.5.b) may belong to some class other than these siblings and the above similarity is not good in making either decision or rejection. We need a new dissimilarity (or similarity) measure for decision making which is suitable to the fuzzy logic search. This measure should be defined for each terminal and consider all pattern classes whose patterns "possibly" reach this terminal with MV above some level. This can be solved in the following way. These classes are first collected for this terminal. Select a group of features which have the best discriminating power among all the possible choices. The dissimilarity (or

similarity) for this terminal can then be defined according to this group of features. In collecting these possible classes, the fuzzy logic search is again useful.

The conventional decision tree is characterized by a series of local decision makings. In collecting all possible classes, the tree classifier is examined globally. The newly defined dissimilarity (or similarity) solves decision making problems globally. We call this process global training, which is decribed in the following algorithm.

ALGORITHM 5.3. Global-Training.

Input : Training patterns.

Output : Decision tree with extended similarity.

1) ISOETRP, which is called recursively and executed interactively to design a decision tree using the given training patterns.

2) Fuzzy logic search, which searches all terminals with MV$\geq\delta$ for each training pattern.

3) Collect the i.d.'s of all training patterns for each terminal node, which reach this terminal in step 2 with MV$\geq\delta$.

4) Select a sub-feature-space for each terminal such that the pattern class indicated in this terminal is

mostly dissimilar to the other train patterns (or classes) collected in step 3.

5) For each terminal, set up an extended node to write the information of this class and this subspace, which at least includes :

feature numbers in this subspace,

class center of this terminal in this subspace.

6) Output the modified tree.

END

This algorithm is actually implemented in many programs. The above gives only its outline.

By the above three improvements, the decision tree becomes more perfect than before and it needs the following search algorithm in the recognition phase.

ALGORITHM 5.4. Fuzzy-Logic-Search-2.

1) Find out a terminal node by straight forward search.

2) If the extended dissimilarity between the unknown pattern and the indicated class is smaller than some threshold, then output the recognition result and terminate the process, otherwise go to step 3.

3) Initialize the potential list LP with only the root

node in it, the MV value being 1. Initialize the terminal list LT as empty.

4) If LP is not empty, then go to step 5

else if LT is empty then output recognition result and terminate

else

begin

For each tepminal in LT do

calculate the extended dissimilarity between X and the indicated class;

Select the class with least dissimilarity;

If this dissimilarity is small enough,

then output the recognition result

else output rejection;

terminate

end;

5) Take off the first node from LP, for each of its child nodes, calculate the $MV_i$ value

$$MV_{ci} = MV^{\mu_i}$$

using eqns.(5.8) and (5.9).

6) Put all child nodes with $MV_i > \delta$ in LP list (for internal nodes) or LT list (for terminal nodes).

Go to step 4.

END

As explained earlier the search guiding function of MV and decision making function of extended similarity are well distinguished in this algorithm. In addition, the fuzzy membership function is given as in eqns.(5.8) and (5.9), which is more related to the information about the region occupied by the pattern class than that given in eqn.(5.3). An experiment on Algorithm 5.3 and Algorithm 5.4 was conducted with very encouraging results. It is stated in Chapter 6, where most principles in both design phase and recognition phase proposed in Chapter 2 are justified.

## 5.4. Conclusion : An Improved Model of Tree Classifier

A decision tree is understood as in Section 1.3 with the straight forward search in the recognition phase. Its main advantage is the high speed of recognition. Its main problems are overlap and error accumulation. The former is solved in Chapter 3 by ISOETRP. To solve the latter, some improvements are given in this chapter, after which the decision tree is no longer understood as before. The design phase of the old decision tree is completed by clustering as in Fig 5.6. The new tree needs one more stage in the design phase, which is the global training as depicted in Fig.5.7. This corresponds to steps 2-5 in Algorithm 5.3. The recognition phase of the new tree classifier is also different from that of the old tree. This is Algorithm 5.4, as shown in Fig.5.8.

Figs.5.6 and 5.8 show us the new model decision tree, which no longer suffers from the error accumulation problem. The following points about the new model constitute the main improvements :

1) To overcome the error accumulation, some search technique other than straight forward one is necessary. While the latter only finds one candidate, the former should find a group of classes which are similar to the unknown in one way or another. The old model uses only local information at each level of the tree. The essential

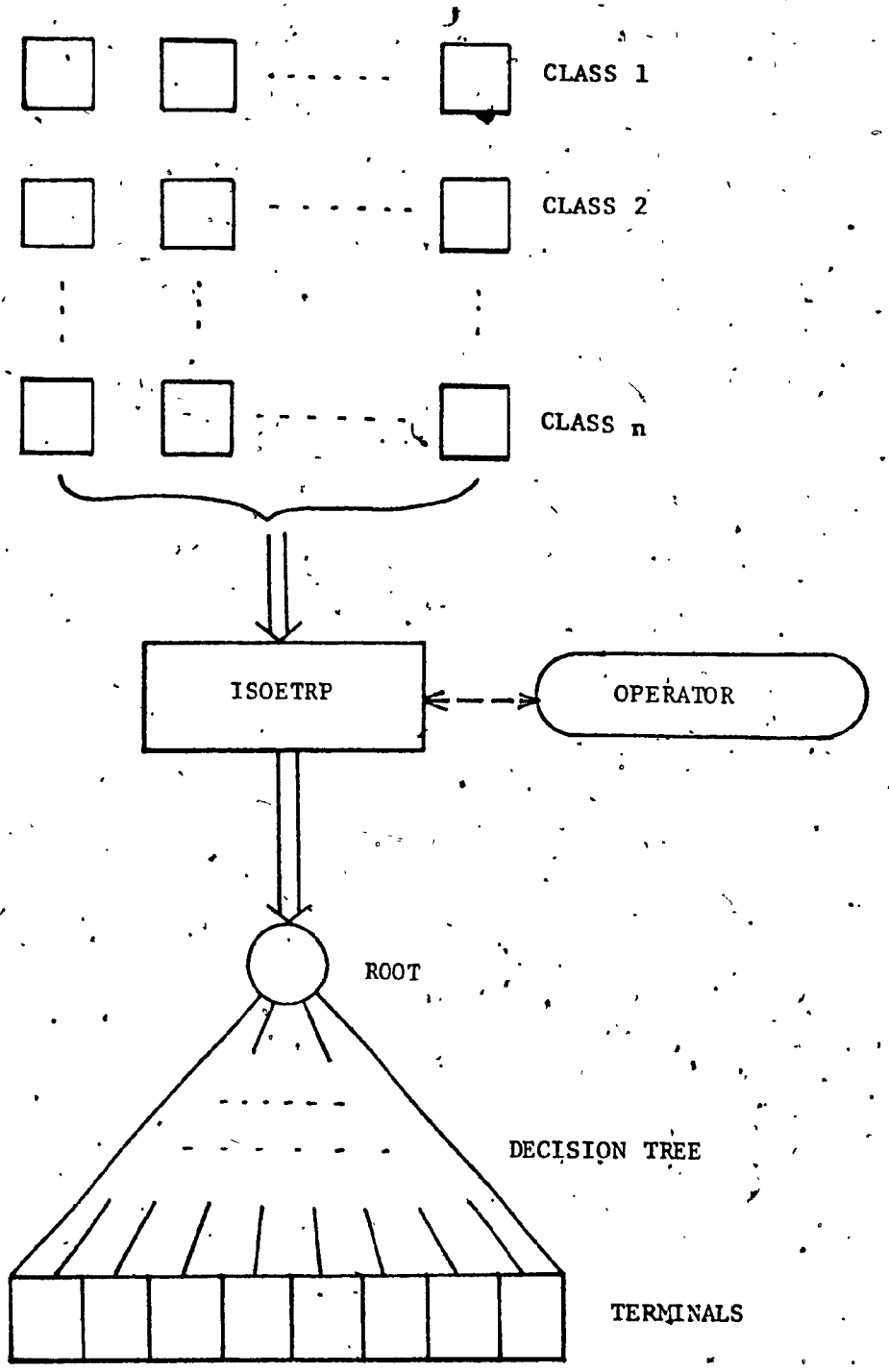



Fig. 5,6 Clustering

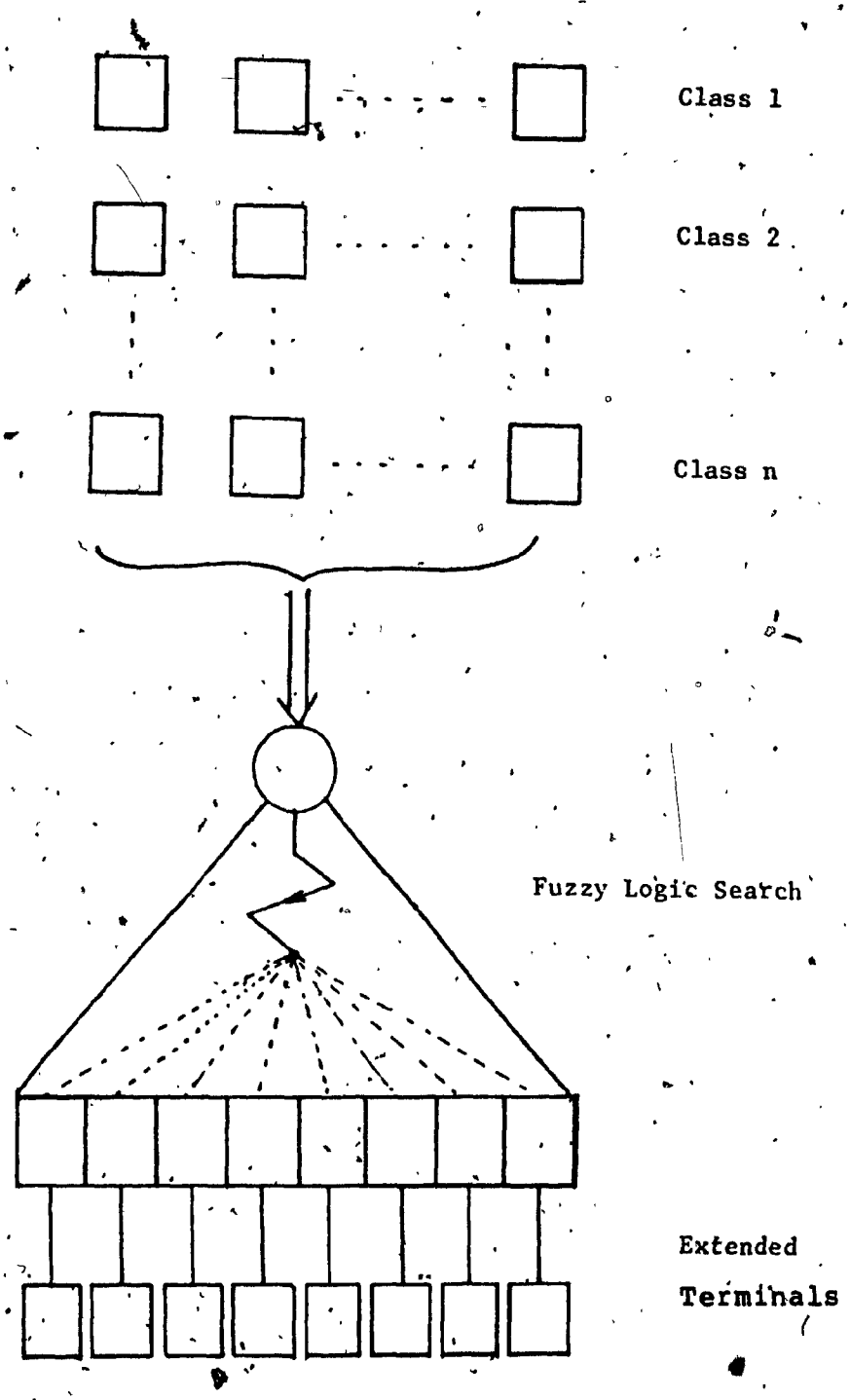



Fig. 5.7 Global Training

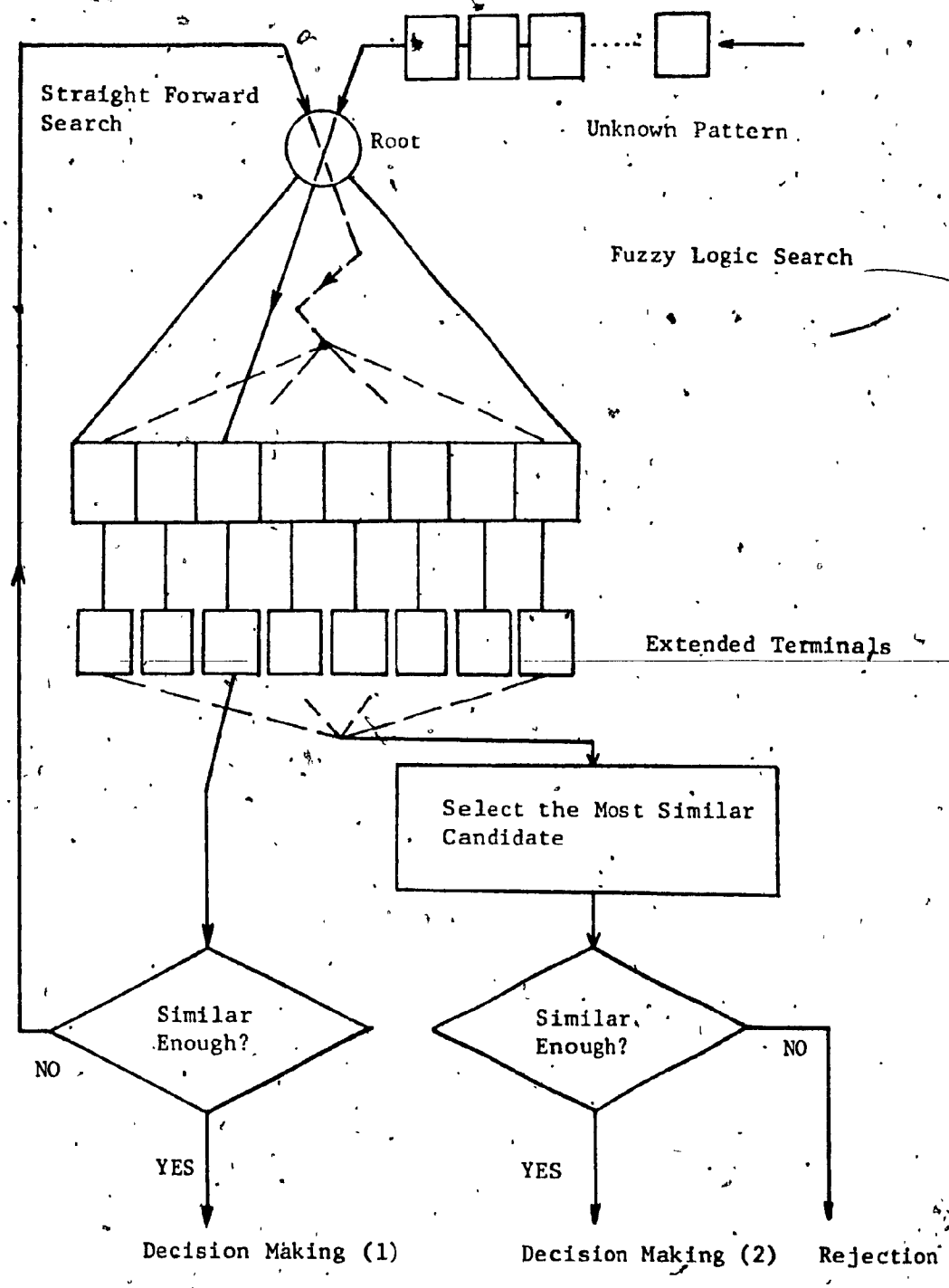



Fig. 5.8 Recognition Phase

improvement in the new model is to record "historical" information, and not lose the way by only one local mistake.

2) The fuzzy membership value under the multiplying rule is an entity for recording this historical information. There should exist other ways, whenever a proper heuristic evaluation is provided, which guides the search. A good choice of the fuzzy membership function is to make it closely related to the occupied region of the pattern class.

3) To switch from recognition to rejection, or from straight forward search to fuzzy logic search whatever, a specified dissimilarity or similarity measure is necessary in each terminal node. In looking for it, we should keep in mind that a decision tree is different from an ordinary classifier. For each terminal node, there is a group of reachable classes. The similarity or dissimilarity measure should be defined among this group aiming at examining the candidate class in this terminal. This reachable class group can be found by fuzzy logic search, as in Algorithm 5.3.

4) The complexity of the new model is well justified, not only because the error rate has been suppressed, but also it provides the kind of flexibility which can not be found in the old model. By adjusting $\lambda$ in eqn.(5.7),

each region $\Omega_i$ which is assumed to be occupied by class i is actually changed. Usually the clustering part takes much more time than the training part (Figs.5.6 and 5.7), and the above adjustment is not easy to achieve in the old model. By adjusting $\delta$ in the search algorithm, the speed-error rate tradeoff is possible. By adjusting the dissimilarity threshold, the error-rejection trade off is possible. Furthermore, the $\epsilon$ value in eqns. (5.8) and (5.9) can also be adjusted, so that the fuzzy heuristic fits the pattern distributions well. All these flexibilities are out of reach in the old decision tree model.

## 5.5. References


[1]. N. J. Nilsson, Principles of Artificial Intelligence, Tioga Publishing Co., California, 1980.

[2]. R. L. P. Chang and T. Pavlidis, "Fuzzy decision tree algorithms," IEEE Trans. on Syst. Man and Cybern., vol. SMC-7, No.1, 28-35, Jan. 1977.

[3]. E. L. Lawler and D. E. Wood, "Branch-and-bound methods : A survey," Operations Research, vol.14, 699-719, 1965.

[4]. L. A. Zadeh, "Fuzzy sets," Information and Control, vol.8, 338-353, 1965.

[5]. L. A. Zadeh, "Calculus of fuzzy restrictions," in Fuzzy Sets and Their Applications to Cognition and Decision Processes, Academic Press, Inc., New York, pp.1-39, 1975.

[6]. Gregg C. Oden, "Fuzzy propositional approach to psycholinguistic problems : An application of fuzzy set theory in cognitive science," in Advances in Fuzzy Theory and Applications, M. M. Gupta, R. K. Ragade and R. R. Yager (eds.), North-Holland Publishing Company, pp.109-420, 1979.

# CHAPTER SIX

# EXPERIMENTS ON COMPUTER RECOGNITION OF CHINESE CHARACTERS

## INTRODUCTION

Several thousand Chinese characters were used as the set of pattern classes in the simulation experiments on tree classifiers. In the ascending order of merit, the design techniques used are histogram method, ISOETRP, interactive ISOETRP and global training (Algorithm 5.3). The design principles based on the analysis in Chapter 2 were applied. The new model of tree classifier is proved to be an important development by the corresponding experiment. In simulating the new model tree classifier, 3200 character classes were used, 99.93% recognition rate was achieved, and the error rate was only 0.025%, at a speed of 861 samples/sec., when running the programs written in Pascal on a CYBER-172 computer.

## 6.1. General Descriptions

The Chinese character set is suitable for the simulation experiments on what have been obtained in the previous chapters. The raw data were obtained by scanning thousands of printed characters and storing them on magnetic tapes as 0-1 matrices. According to ref. [1.19], the frequency of occurrence of the 3072 most commonly used characters indicates that they account for more than 99.7% of contemporary Chinese. The number of character categories in each experiment varied from 200 to 3200. All simulations were written in a high level language, Pascal, and performed on a Cyber-172 computer at Concordia University.

1) Noise Models

For simulation purpose, a noise model has to be used to generate noise samples either for classifier training or testing. The model is not intended for industrial use, but the noise in the sample should not be too easy to eliminate, so that the classifier can be tested to see if it can recognize noisy characters.

Two noise models were analyzed in work [1.14]. The first is a white noise model. Suppose A is the 0-1 matrix corresponding to a character. Let us set another matrix B in such a way that

$$B(i,j) = \text{Random for each } (i,j), \qquad (6.1)$$

which is uniformly distributed in the interval [0,1). Noise is added to different points (i,j) of A such that

$$A(i,j) \text{ is changed either from 0 to 1,}$$

$$\text{or from 1 to 0} \qquad (6.2)$$

where (i,j) is determined by a 0-1 noise matrix N. The value of the elements of N are determined by matrix B :

$$N(i,j) = 1, \text{ if } B(i,j) \leq \alpha$$

$$0, \text{ otherwise} \qquad (6.3)$$

where $\alpha$ is a predefined threshold reflecting the noise level. Examples of noisy characters generated in this model are shown in Fig. 6.1b.

It is easy to compute the expected value (EN) and variance (DN) and correlation function (R) of the noise matrix N :

$$EN(I,J) = \alpha$$

$$DN(I,J) = \alpha(1-\alpha)$$

$$R(i,j) = \alpha(1-\alpha), \text{if } (i,j) = (0, 0)$$

$$0, \text{ otherwise} \qquad (6.4)$$

Most noise points in this model are isolated from each

a.



b.

Fig. 6.1 Noise models

a. model specified in Eqn. 6.5.,

b, white noise model.

other, as seen in Fig. 6.1b, hence they can be easily removed by simple preprocessing techniques.

The second model was proposed in [1.14], which is

$$N(i,j) = \begin{cases} 1, & \text{if } t \leq \alpha \\ 0, & \text{otherwise} \end{cases}$$

where

$$t = \beta B(i,j)+B(i-1,j)+B(i+1,j)+B(i-1,j-1)+B(i,j-1) +B(i+1,j-1)+B(i-1,j+1)+B(i,j+1)+B(i+1,j+1) \tag{6.5}$$

and $\beta > 1$ is a suitable constant. Eqn. (6.5) means that if point (i, j) is a noise point, then all its neighbors are also likely to be noise points. This corresponds to the real noise situation produced by the printing process.

The correlation of this random field, N, can be computed as follows. Let points (i-1, j-1), (i, j-1), ..., (i+1, j+1) be denoted by 1, 2, ..., 9, as shown in Fig.2, then we have

$$t = \beta B_5 + \sum_{i=2}^{9} B_i$$

and

$$E(N_5) = P\{N_5 = 1\}$$

| | | | |
|---|---|---|---|
| 1 | 4 | 7 | 10 |
| 2 | 5 | 8 | 11 |
| 3 | 6 | 9 | 12 |

Fig. 6.2 Brief Index

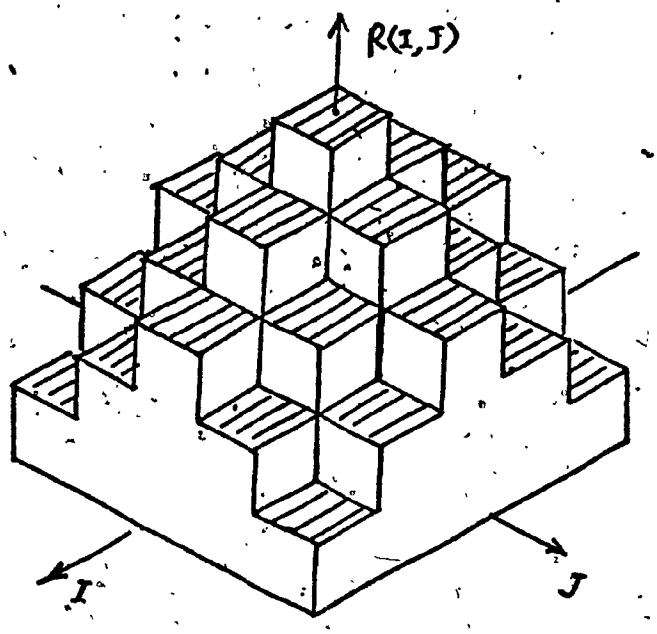



Fig. 6.3 Non-White Noise Model

Discrete Correlation Function

$$= P\{t \leq \alpha\}$$

$$= \int_0^{\alpha} dt \int_0^{u_1} du_1 \int_0^{u_2} du_2 \cdots \int_0^{u_7/\beta} d\beta_5$$

$$= \int_0^{\alpha} t^8/(8!\beta)\, dt$$

or

$$E(N_5) = \alpha^9 / (9!\beta) \qquad (6.6)$$

Let

$$U = B_1 + B_2 + B_3 + (\beta-1)B_5$$

$$V = B_{10} + B_{11} + B_{12} + (\beta-1)B_8$$

$$W = B_4 + B_7 + B_6 + B_9 + B_5 + B_8$$

then

$$E(N_5N_8) = P\{(N_5=1) \wedge (N_8=1)\}$$

$$= P\{(U+W) \leq \alpha \wedge (W+V) \leq \alpha\}$$

$$= \int_0^{\alpha} dt \int_{t-\alpha}^{t} du \int_0^{\alpha+u-1} dV$$

$$= \alpha^9 / (8!(\beta-1)^2) \qquad (6.7)$$

Hence

$$R(0,1) = R(-1,0) = R(0,-1) = R(1,0)$$

$$= E(N_5N_8) - EN_5EN_8$$

$$= \alpha^9/(8!(\beta-1)^2) - (\alpha^9/(9!\beta))^2 \qquad (6.8)$$

This correlation function $R(i,j)$ is depicted in Fig. 6.3.

The noise level, as well as the correlation function can be controlled by adjusting the parameters $\alpha$ and $\beta$. By calculation, we have

$$\partial R(0,1)/\partial\beta = \frac{-2}{\beta-1}\,\alpha^9/(8!(\beta-1)^2) - \frac{-2}{\beta}\,(\alpha^9/(9!\beta))^2$$

$$= -2\,[R(0,1)/\beta + \alpha^9/(8!\beta(\beta-1)^3)]$$

$$< 0. \qquad (6.9)$$

which means that, the larger $\beta$ is, the smaller the correlation $R(0,1)$ is. When $\beta \longrightarrow \infty$ ($\alpha$ varies correspondingly), this model becomes a white noise model.

Model (6.5) is a non-white noise model, which was used in all the simulations described in this chapter, for generating training as well as testing samples. Examples of noisily generated characters under this model are shown in Fig. 6.1a. We can see, noise points tend to lump together. A simple low pass filter will not be able to eliminate noise produced by this model.

2) Positioning Method

It is important to locate the character in the matrix so that its center has a stable position under noise. The

positioning method should suppress the noise effect on features such as Walsh transforms and phase features. The following presents some positioning methods on profiles

$$X(i),\ i = 1, 2, \ldots, N$$

as shown in Fig. 6.4. We define the symmetric center of the profile as a point S, such that

$$\sum_{i<[S]} X(i) + (S-[S])\, X([S]) = \sum_{i>[S]} X(i) + ([S+1]-S)\, X([S]) \qquad (6.10)$$

where [S] is the integer part of S.

We can also define the mass center as

$$M = \sum_i I\, X(i) \,/\, \sum_i X(i) \qquad (6.11)$$

S and M are real numbers. When S is used, the profile is tranlated to a new position

$$X'(i) = (S-[S])\, X(i+[S+1]-N/2) + ([S+1]-S)\, X(i + [S]-N/2). \qquad (6.12)$$

M is used in a similar way. In addition we can define

$$M_r = \text{Round}(M)$$

$$S_r = \text{Round}(S)$$

and similarly, (6.12) becomes

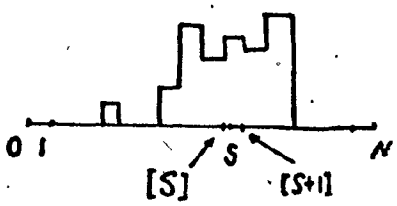



Fig. 6.4 Symmetric Center of the Profile

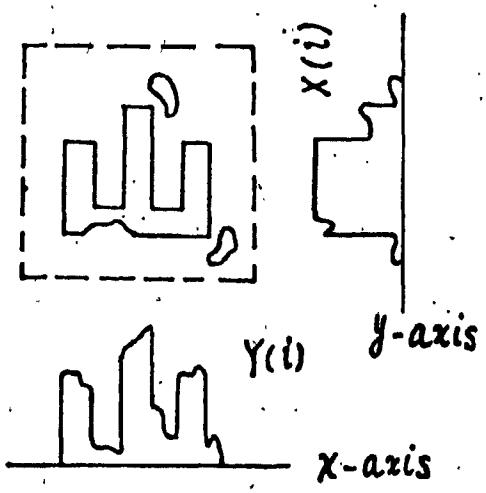



Fig. 6.5 The X-Y profiles of a character.

$$X'(i) = X(i-S_r-N/2).$$

Positioning methods by assigning S, M, $S_r$ or $M_r$ as the center of the profiles were compared using about 100 digitized noisy characters. Performance measures of the Walsh transforms for the 4 cases are shown in Table 6.1, where one sees that S and M are better than $S_r$ and $M_r$, and M is slightly better than S.

3) Feature Selection

Suppose some feature evaluation criterion is chosen and d' features are selected out of the set of d features, then there are $\binom{d}{d'}$ candidate subsets. It has been proved that the search for the optimal subset in the general case has to be exhaustive [6.3, 6.4]. The branch-bound algorithm has been reported in [6.5] to reduce the amount of enumeration. But still too much time is needed, which makes the seeking of the optimal subset impractical. Many suboptimal search methods have been invented, as given in a survey [6.6]. But in our tree classifier for thousands of characters, feature selection must be done for each internal node, totally hundreds or even thousands of times. In the design process, we need an efficient method.

A tree classifier is different from a one stage classifier in that the former only makes local decisions at each level. Since the recognition rate depends on all the levels of decisions, many features are actually used in the whole

TABLE 6.1

Performance of the Features Using 4 Centralizing Methods

| | The 3 Best Ones | | | AVERAGE |
|---|---|---|---|---|
| | 1 | 2 | 3 | |
| S | 6.38 | 5.38 | 5.06 | 3.45 |
| G | 7.63 | 7.35 | 5.81 | 3.59 |
| $S_r$ | 6.75 | 5.19 | 4.88 | 2.65 |
| $G_r$ | 7.31 | 5.69 | 5.25 | 2.71 |

process. The features which have been used in higher levels of the tree usually become less informative than those unused before. By our experience a good feature will usually be used sooner or later in the whole design process, even a simple selection method is used.

Such transforms as fast Walsh, FFT were used in all the simulations. In this way, the features obtained reflect the global property of the character. They are not sensitive to noise [6.7]. It has also been pointed out [4.12] that, after orthonormal transformation, the features become less related than those of the raw pattern. Under this condition, we can use the simple method in which individual best features are selected. This method selects d' best features out of d features, and only d times of feature measuring are necessary, which is practical in the large tree design. Although this approach may not be optimal, it works well in the design of the tree classifier when orthonormal transforms are used.

The feature evaluation criterion used is the scatter ratio, or Fisher's criterion given in eqn. (4.35).

4) Programs

Owing to the huge data size, quite a lot of programs were used in a single experiment. But the following programs are usually the important ones common to all the simulations :

--FEATURE EXTRACTION :

Noise pattern generating.

Profile, mesh or other local feature creation.

FFT, Walsh or other orthonormal transforms.

-- TREE DESIGN :

Feature selection.

Histogram method or ISOETRP clustering.

Tree created is stored in a disk file.

--GLOBAL TRAINING of the TREE :

Find all terminals each pattern can reach.

Collect reachable patterns for each terminal.

Select sub-feature-space for each terminal.

Modify the tree with extended similarity measure.

--TESTING:

Tree recreation in memory (from disk file).

Straight forward search.

Fuzzy logic searches.

## 6.2. Design with Histograms

The decision tree dealing with 3155 characters was simulated in July, 1981 [1.14]. The features used are the Walsh coefficients extracted from two profiles of a character projected onto the X-Y orthogonal axes (Fig. 6.5).

$$X(i) = \sum_{j} A(i,j)$$

$$Y(j) = \sum_{i} A(i,j)$$

where A(i,j) is the 0-1 matrix corresponding to the character pattern.

Each class is considered to have a multivariate normal distribution in the feature space. These features are assumed to be independent. The variances of the features were estimated by machine generated noisy samples, 10 for each class. The pattern class is assumed to occupy the region

$$(\mu - \alpha\,\sigma,\ \mu + \alpha\,\sigma)$$

along each feature axis, $\mu$ being the mean value and $\sigma$ the variance. $\alpha$ was chosen as 3.25 (see Table 2.1).

Only one feature is used in each internal node of the tree and the discriminant is simply an inequality. Histograms were used in the design. The program is a recursive one. At first there is only one internal node

(root) in the memory, which contains all 3155 characters. For this unprocessed node, the following are done :

1) Select the single best feature.

2) Print the histogram of the character classes along this feature axis.

3) The operator selects several points (cut) to divide the axis into segments such that pattern classes are separated into smaller groups, each corresponding to a segment. The overlap is determined according to $\alpha = 3.25$. The principle of "cut" point selection is to try 2--6 segments and meanwhile control the overlap between the new groups.

4) Each smaller group is assigned to a child node.

5) For each child node with more than one pattern class in it, repeat the above steps.

6) The process continues until all nodes with more than one class have been processed.

7) The result is a decision tree in the memory, which is then written to a disk file.

This tree contains about 15000 overlaps. On the average, the terminal nodes sit at level 10, but the height of the tree is 20.

Testing samples were prepared in the same way as training samples, but independent to the latter. 9345 testing patterns were used (3 for each class). The recognition rate is 99.5%. The speed is about 50 characters per CPU second when the recognition experiment was carried out on the CYBER 172 computer. Some examples of correctly recognized and misrecognized characters are shown in Figs.6.6 and 6.7 respectively, where we can see even very noisy characters were correctly recognized by this classifier. Notice that some others with the same noise level were misrecognized which could be recognized if we had used more than one feature in each internal node, or if we had used a better search method, such as fuzzy logic search.

The experiment is the first in the series of simulations for the thesis. The main shortcomings of this tree are : serious overlap, which uses up too much memory. Speed is not satisfactory. There is no rejection option in the recognition result, and some of the misrecognized characters shown in Fig. 6.7 might have been rejected, which would be preferred.

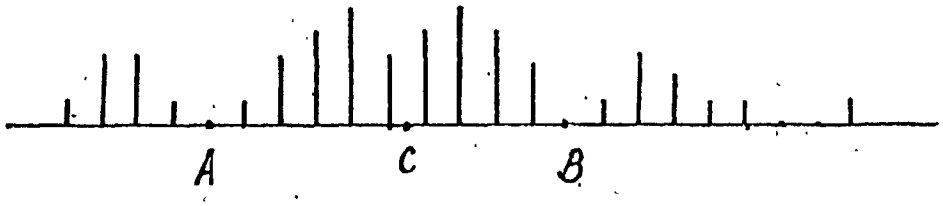



Fig. 6.6 An Example of Histogram for Interactive Design

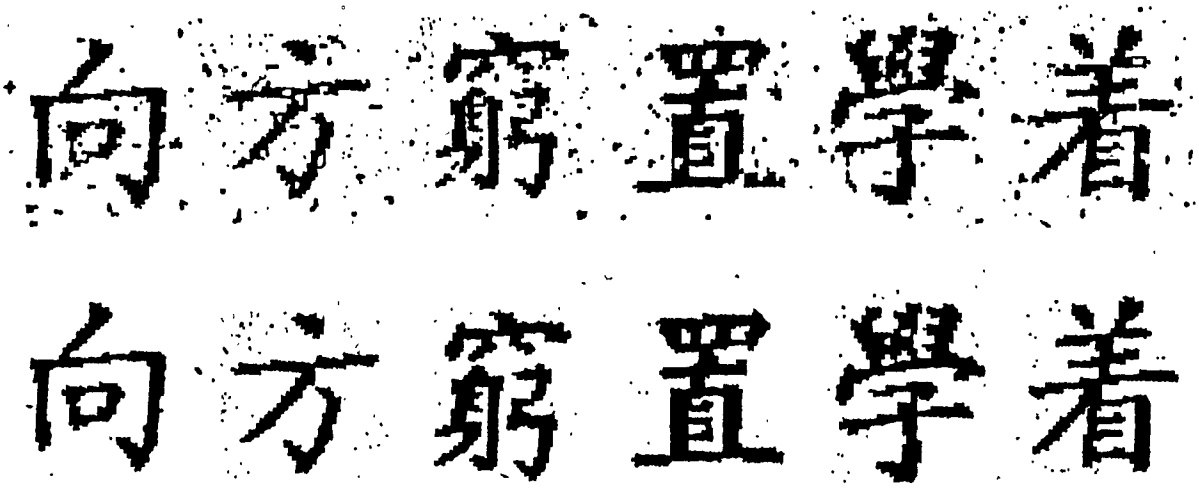


Fig. 6.7a Examples of Correctly Recognized Characters

(With Reference to Clean Patterns)

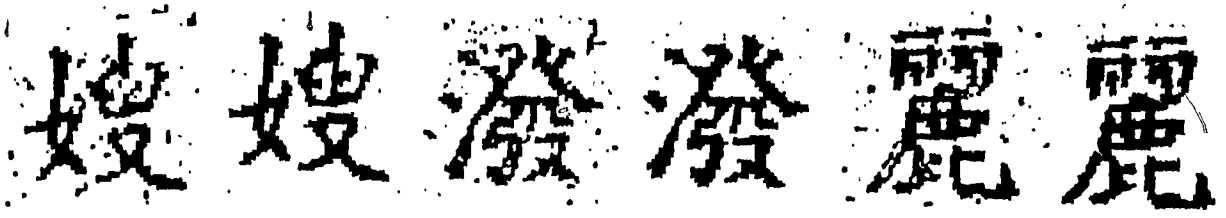


Fig. 6.7b Examples of Mis-recognized Characters.

(With Reference to Clean Patterns)

## 6.3. Design with ISOETRP

A tree classifying 500 characters in multifont was simulated in July, 1982 [3.11].

Four different fonts were generated from a single font of characters. The variance of stroke width among these fonts is evident as seen in Fig. 6.8. In addition to random noise and shift noise, rotation noise was added.

The features used are crossing counts and shades, as illustrated in Fig. 6.9. Crossing counts are prepared as follows. The right and lower edges of each stroke are extracted, resulting in two patterns. Both the right edge pattern and lower edge pattern are projected in horizontal and vertical directions to produce 4 curves. To prepare shade features, 4 side patterns of the character are extracted, they are the black areas each occupying 20% of the total black area. These side patterns are then projected in corresponding directions (Fig. 6.9) to produce the other 4 curves. These 8 curves are divided into 4 groups, each of which containing one shade and one crossing count in the same direction. Fourier transformation was applied to these curves to obtain 4 groups of numerical features, each of which contains 15 (amplitude) Fourier coefficients, 8 from the shade curve and 7 from the crossing count curve, where the integration coefficient of the

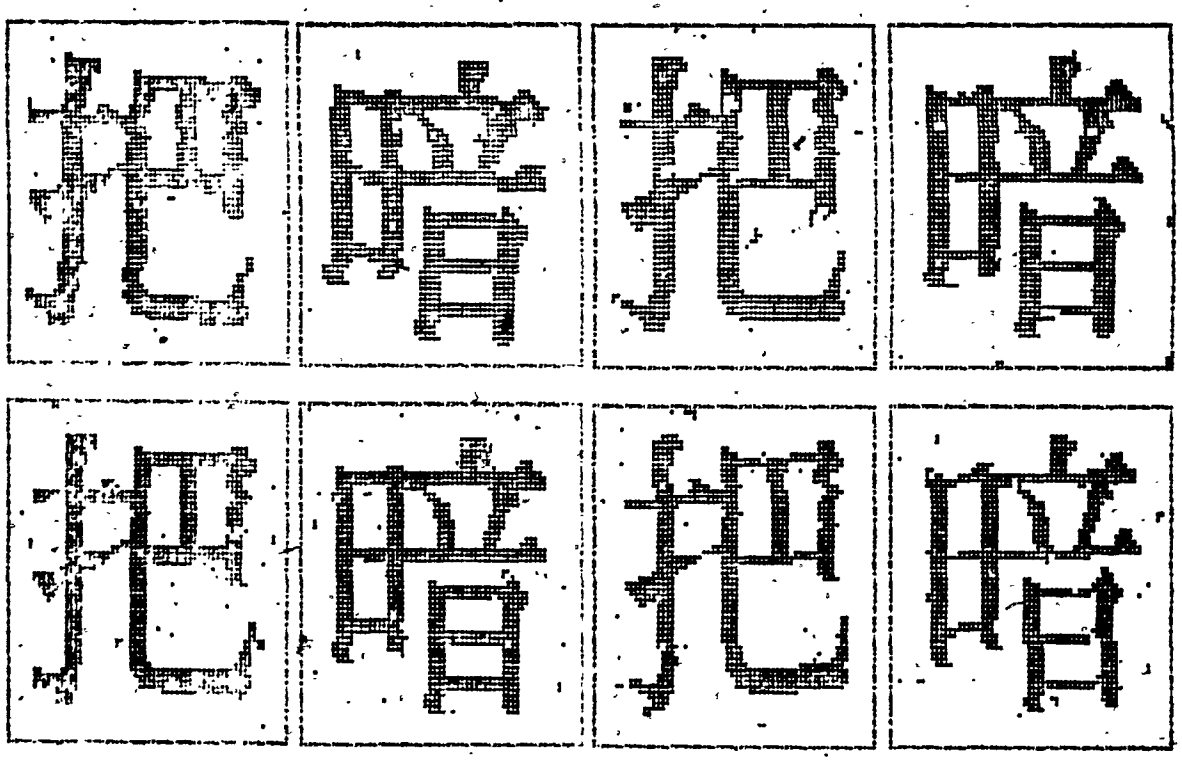


Fig. 6.8 Character samples used in the simulation.

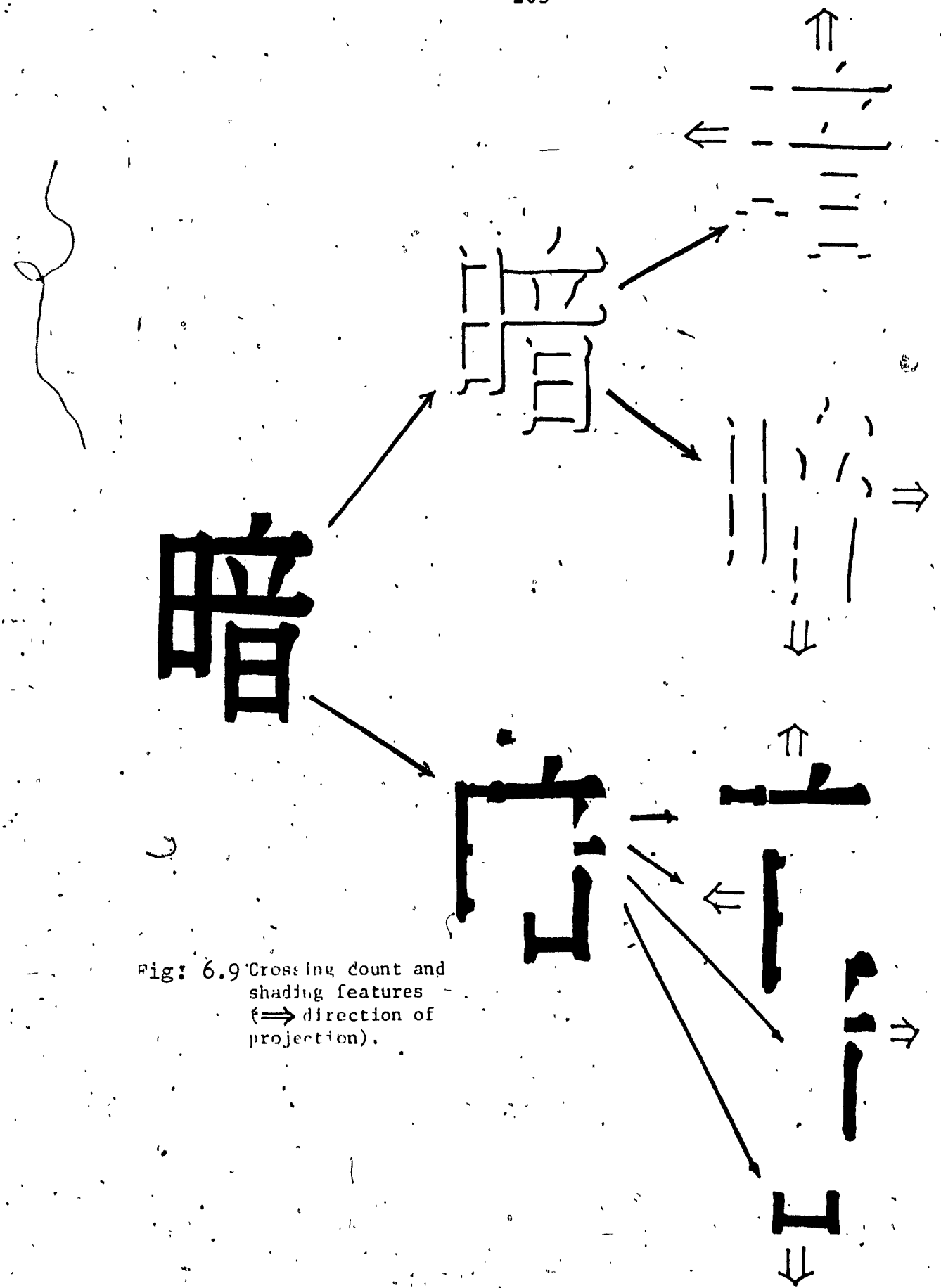


Fig: 6.9 Crossing count and shading features (⟹ direction of projection).

crossing count curve is not counted in the 15. In addition, 3 more features are extracted : the total number of black points, the integration of vertical and horizontal crossing count curves. In this way 4 groups of numerical features are formed corresponding to 4 different directions. Intuitively, they are independent of each other, which makes feature selection easier : one best feature is chosen from each of the 4 groups. These 4 features together with the 3 additional features form a candidate group, from which 4 best features are selected. The feature ordering measure for feature selection is as follows,

$$J = \sigma \; / \; (\sum_{i=1}^{n} \sigma_i / n) \qquad (6.13)$$

where $\sigma_i$ is the variance of the corresponding feature among the ith pattern class, and $\sigma$ the variance over all pattern classes. Table 1.2 shows this measure of the features under various noise conditions or distortions, where we can see both the crossing counts and shades remain stable in random noise. Furthermore the crossing count feature is more stable than the shade feature when the character samples are tilted.

In ref.[1.14], the experiment showed that, when the noise is random, the distribution of each simple font is much like Gaussian. In the case of multi-font, the distribution of each pattern class can be considered as

$$p(x) = \sum_{k=1}^{l} P_k \; p_k(x) \qquad (6.14)$$

where $P_k$ and $p_k(x)$ are the a-priori probability and distribution density of the k-th font respectively. Moreover, distributions under stroke width variation or rotation of the patterns are intrinsic rather than random. In this case eqn. (6.14) is also applicable with the modification that k = 1, 2, ..., l for different fonts as well as the possible intrinsic distortions as explained in Fig. 6.10. If the feature is good, then the local maxima in the distribution of the same pattern classes tend to be close to each other, otherwise they are far apart.

The uninteractive version of ISOETRP was employed in the tree design. The input data contain the information of each pattern class, such as the center of this class and the diameter of the region it covers in the feature space. Initially all classes are considered as in a single cluster. The output data consist of several clusters, each of which containing a number of classes. In the next iteration, this output will serve as input and ISOETRP will be applied to the data of each cluster individually. After a number of iterations (or the number of levels) the tree is created as a file in the secondary memory.

2000 training samples, 4 for each class, were used to design the tree. 2000 independent samples were used to test the tree. This tree of 500 characters in 4 fonts has about 2000 overlaps. The recognition rate is 100% for the

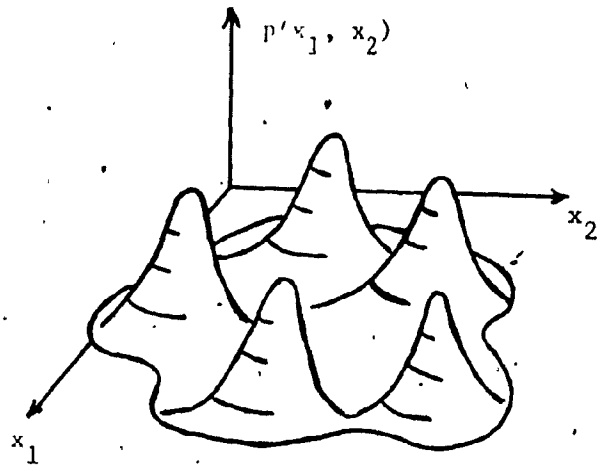



Fig. 6.10 Multi-font distribution model for one character.

Tab. 6.2 Feature Performance Measure under Various Noise Conditions

| Conditions \ Average Separability | $\bar{J}_1$ (Total) | $\bar{J}_1$ (Crossing Count) | $\bar{J}_1$ (Shade) |
|---|---|---|---|
| After thickness variation | 7.280 | 7.723 | 6.267 |
| Masking by random noise | 5.710 | 5.590 | 5.534 |
| $1^\circ$ rotation | 4.546 | 4.930 | 3.912 |
| $2^\circ$ rotation | 3.741 | 4.792 | 2.457 |

training sample set, and 98.5% for the independent testing sample set. The recognition speed is 50 characters/sec, when run on a Cyber 172 computer.

This experiment shows that multifont character recognition by tree classifier are possible, although the data are more difficult to handle than those in the single font problem. The samples were generated by software, which may not reflect the real case. For example, after rotation, filling and smoothing were then applied to the patterns resulting in a lot of additional distortion. It is possible ISOETRP in its uninteractive form does not work very well for some level or internal nodes of the tree, which degrades the tree classifier performance. Anyhow it makes higher dimensional clustering possible in the tree classifier design.

## 6.4. Interactive ISOETRP and Fuzzy Logic Search

Two experiments were conducted using 200 and 1000 frequently used Chinese characters respectively on the Cyber 172 computer at Concordia University. Noisy samples were generated, 20 for each pattern class, i.e. 10 training samples and 10 other testing samples.

Each sample pattern is put in an 8X8 mesh, and the black area in each small region is calculated, which is called the mesh feature (Fig. 6.11). 2-D FFT was applied to the mesh features. In order to save computing time, a pair of samples were computed at the same time. Let $A_1$ and $A_2$ be the character metrices, then FFT was applied to the complex matrix

$$B = i A_1 + A_2$$

$$B \leftarrow FFT(B)$$

where i is the imaginary unit. After FFT, the features of $A_1$ and $A_2$ were separated using the following formula

$$u = (n - i) \bmod n$$

$$v = (n - j) \bmod n$$

$$F_1(i, j) = Re(B(i, j)) + Re(B(u, v))$$

$$F_2(i, j) = Im(B(i, j)) + Im(B(u, v))$$

$$F_1(u, v) = Im(B(i, j)) - Im(B(u, v))$$

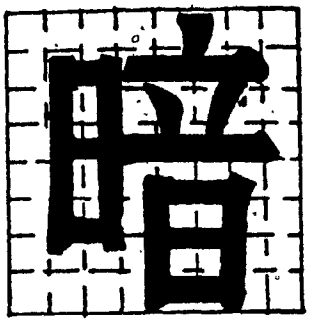


Fig. 6.11 Pattern Matrix

TABLE 6.3 Decision Trees

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| No. of Categories | 200 | | | | 1000 | | | |
| No. of internal nodes | 84 | | | | 477 | | | |
| No. of terminals | 205 | | | | 1205 | | | |
| Overlap | 5 | | | | 205 | | | |
| Terminals and their levels | 17 | 158 | 30 | | 40 | 477 | 630 | 58 |
| | 3 | 4 | 5 | | 4 | 5 | 6 | 7 |
| Average level of terminals | 4.112 | | | | 5.586 | | | |
| Branch factor | 3.65 | | | | 3.56 | | | |

$$F_2(u, v) = Re(B(u, v)) - Re(B(i, j)) \qquad (6.15)$$

where $F_1$ is the feature of $A_1$, and $F_2$ of $A_2$. Then magnitude and phase features were computed as in Section 4.3.

After the data had been prepared, the interactive version of ISOETRP was used to design the tree classifier.

One tree for 200 classes and the other for 1000 classes were designed. They are shown in Table 6.3. On the average, the terminal nodes in these two trees sit between levels 4 and 5. The branch factors are 3.65 and 3.56 respectively, which were computed by

$$B = \exp((\ln n)/\bar{L}) \qquad (6.16)$$

where n is the number of terminal nodes and $\bar{L}$ the average level of them. This coincides with eqn. (2.45), one of the design principles.

In testing the tree, both the straight forward search (Algorithm 5.1), and fuzzy logic search (Algorithm 5.2), were applied and the results are shown in Table 6.4. From these results, the following can be seen :

1) The average number of terminal nodes accessed by the testing patterns in these two trees ranges from 16.685 to 22.326, which is very close to the value of $B^{\bar{L}}$. This justifies that the time required for each unknown in a tree classifier is $O(\log n)$ since

TABLE 6.4 Tree Searches

| No. of categories | | 200 | 1000 |
|---|---|---|---|
| I | average no. of nodes accessed | 16.685 | 22.326 |
| | error rate | 0.2% | 1.07% |
| | recognition rate | 99.8% | 98.93% |
| | speed | 416 samples/sec. | 297 samples/sec. |
| II | no. of terminals searched by each backtracking sample | 2.25 | 2.58 |
| | error rate | 0 | 0.26% |
| | rejection rate | 0 | 0.09% |
| | recognition rate | 100% | 99.65% |
| | speed | 415 samples/sec. | 288 samples/sec. |

I. Straight forward search
II. Fuzzy logic search.

$\bar{L} = \log n / \log B$

where B is nearly a constant.

2) The error rate produced by straight forward search is not high, which guarantees that the fuzzy logic search is necessary only for a few samples and will not take up much time. As shown, the recognition speed in fuzzy logic search is only slightly slower than that in the straight forward search.

3) The recognition rate has been improved considerably by the fuzzy logic search, which actually searches only a few more terminal nodes than the straight forward search for a test sample.

4) The overall speed is high considering that these trees were implemented in a high level language environment. A high speed has been achieved for two reasons :

A) The number of nodes accessed by each sample is O(log n), much less than that in a matching scheme.

B) In each node, either an internal or a terminal, a much smaller number of features (4 in this tree) are used than that in a matching scheme. In the latter case (see Section 1.2), each eigen vector contains 64 components as features.

These experiments show that the tree classifier has become a viable approach to the recognition of a large character set, such as Chinese characters, after the development of ISOETRP clustering algorithm and the fuzzy logic search.

## 6.5. Tree Classifier after Global Training

Three experiments were conducted using 64, 450 and 3200 characters respectively. Noisy samples were generated : 15 for each pattern class, 10 being training samples, and 5 testing samples.

Each sample pattern was first put in a 8X8 mesh. 2-D fast Walsh transform was then applied to the mesh features, resulting in 64 features for each pattern.

After data preparation, ISOETRP was used in the same way as described in the last section to design the tree classifiers, to which the global training algorithm (Algorithm 5.3.) was applied. The descriptions of the resulting trees are shown in Table 6.5. Their branch factors, calculated by eqn. (6.16), are in the range required in eqn. (2.45). Notice that in each tree, the number of extended terminals is less than the number of terminals. The reason is as follows. When applying step 2 of Algorithm 5.3, some terminal nodes are reached by only one pattern class, which is just the correct one. In this case, these terminals do not need extensions. The time used to recreate each tree in the memory is written in the same table for reference. The number of internal or terminal nodes sitting at different levels are shown in Table 6.5. The average levels of the trees are 3.0, 4.132 and 5.415 respectively. It is very interesting to note that they

TABLE 6.5 Tree Description

| No. Classes | | 64 | | 450 | | 3200 | |
|---|---|---|---|---|---|---|---|
| | | internal | terminal | internal | terminal | internal | terminal |
| LEVEL | 1 | 5 | 0 | 6 | 0 | 7 | 0 |
| | 2 | 18 | 0 | 29 | 0 | 39 | 0 |
| | 3 | | 64 | 115 | 0 | 222 | 0 |
| | 4 | | | 23 | 418 | 848 | 43 |
| | 5 | | | | 65 | 638 | 2320 |
| | 6 | | | | | 2 | 1746 |
| | 7 | | | | | | 4 |
| No. Internals | | 23 | | 173 | | 1756 | |
| No. Terminals | | 64 | | 483 | | 4113 | |
| Overlaps | | 0 | | 33 | | 913 | |
| Average Level for Terminals | | 3.0 | | 4.132 | | 5.415 | |
| Branch Factor | | 4.0 | | 4.462 | | 4.650 | |
| No. extended terminals | | 48 | | 272 | | 2173 | |
| Recreation time (sec.) | | 0.153 | | 1.096 | | 10.171 | |

nearly form an arithmatic series while the corresponding numbers of classes, being 64, 450 and 3200 respectively, constitute a geometrical series. This justifies the theoretical analysis of the tree classifier presented in Chapter 2.

The results of both the straight forward search and the fuzzy logic search as described in Algorithm 5.4 using independent testing samples are shown in Table 6.6. Especially, the comparison of trees with and without global training was made. Fig. 6.12 shows examples of some characters, which were misrecognized in the straight forward search, but re-recognized in the new model tree. Some testing samples under various noise levels were input to the trees and the results are shown in Fig. 6.13. From these figures we can make the following conclusions :

1) The recognition rate obtained from the straight forward search, although generally high, decreases when the number of classes increases. This is due to error accumulation. This effect is considerably reduced by the fuzzy logic search, as indicated in Table 6.6. When the tree is treated as a new model with sophistication, the error accumulation effect can nearly be overcome. Although the rejection rate is not zero, it is as low as 0.044%. Furthermore, with the rejection option available in the new model, the error rate is as low as 0.025% for the tree for 3200 characters. Some misrecognized

TABLE 6.6 RECOGNITION RESULTS

| Number of Classes | | 64 | 410 | 3200 before global training | 3200 after global training |
|---|---|---|---|---|---|
| I | Error Rate | 0 | 0.3% | 1.4% | |
| | Recognition Rate | 100% | 99.7% | 98.6% | |
| | Speed | 1758/sec. | 1318/sec. | 958/sec. | |
| II | Backtracking rate | 0 | 0.004 | 0.0134 | 0.0146 |
| | Error Rate 1 | 0 | 0 | 0.09375% | 0.01875% |
| | Error Rate 2 | 0 | 0 | 0.01875% | 0.00625% |
| | Error Rate | 0 | 0 | 0.1125% | 0.025% |
| | Rejection Rate | 0 | 0.09% | 0.00625% | 0.044% |
| | Recognition Rate | 100% | 99.91% | 99.88% | 99.93% |
| | Speed | 1730/sec. | 1216/sec. | 873/sec. | 861/sec. |

I : Straight forward search

II : Fuzzy logic search

Error rate 1 : error in step 2) of Algorithm 5.4.

Error rate 2 : error in step 4) of Algorithm 5.4.

Fig. 6.12 XY Each X is misclassified as Y in the conventional search, but re-recognized in the new model tree

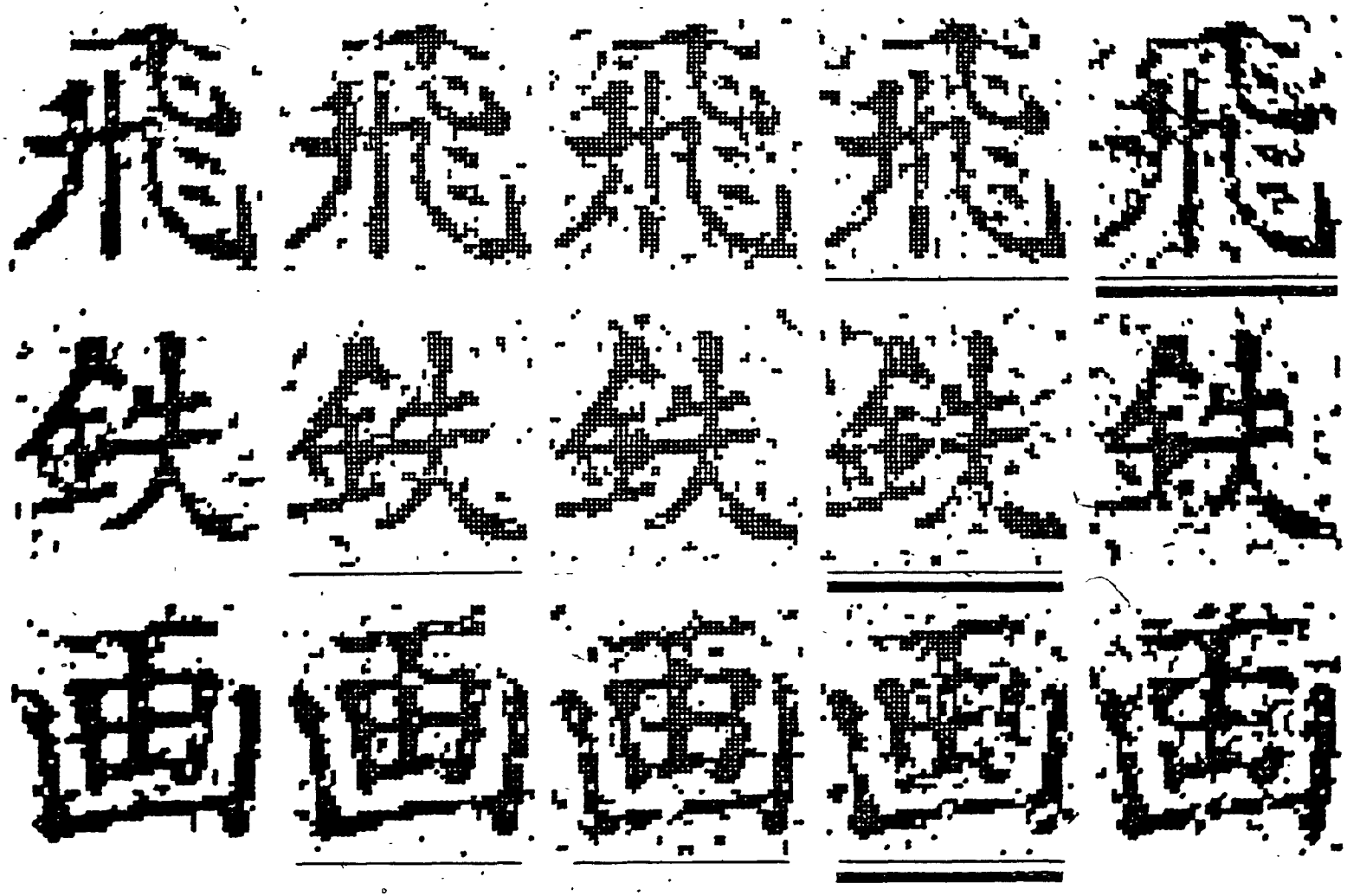
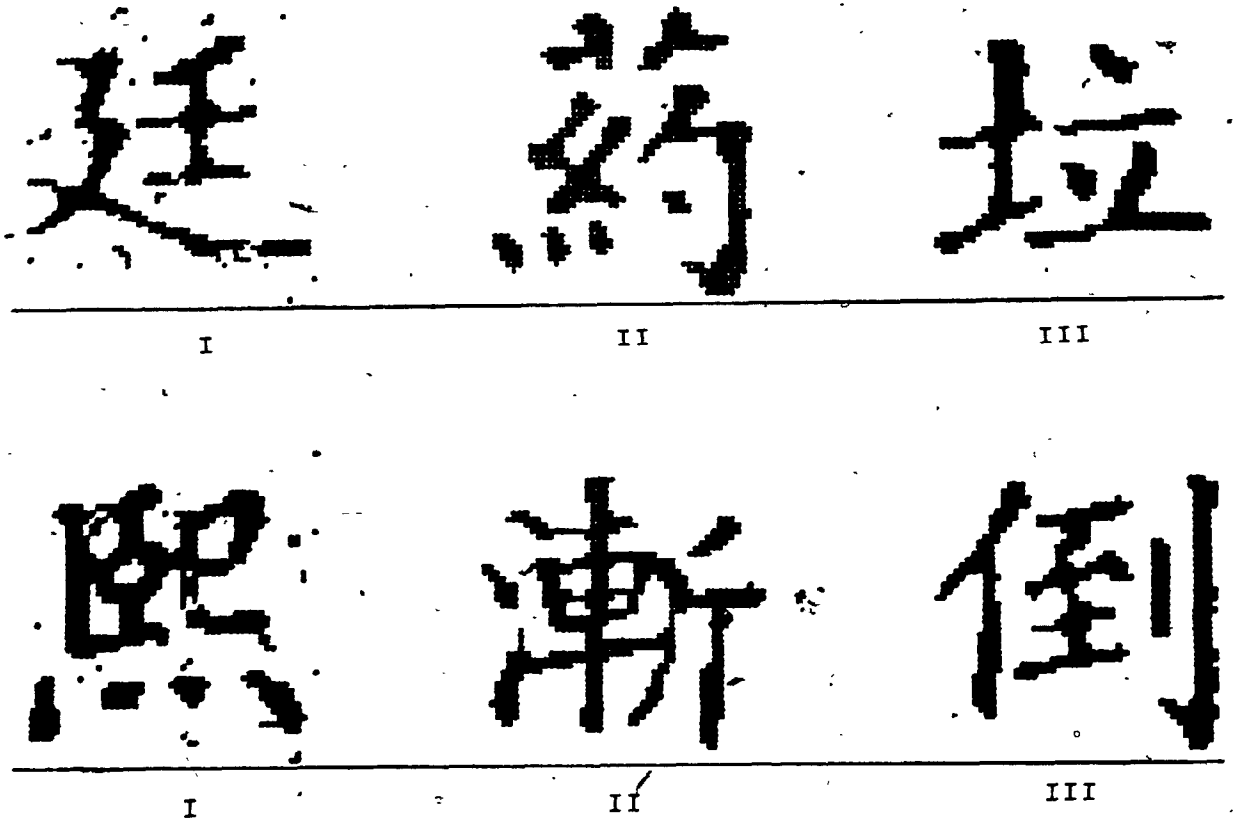


Fig. 6.13 Multi noise level samples, ═ rejected, ── re-recognized, others recognized.

character samples are shown in Fig. 6.14. For example the character " 廷 " was classified as " 葯 " in the first search, and the result of the second search is " 垃 ", which is still a mistake. Character " 廷 " is neither similiar to " 葯 ", nor to " 垃 " from human's sense and understanding. The reason is that the features used in the tree classifier simulation were numerical ones, instead of structural features. But still, by globally training the tree classifier and adding the fuzzy logic search, the error rate can be suppressed to an acceptable level. In this new model, it is possible to re-recognize those misclassified patterns as in Fig. 6.14, which will be explained in part 3). In the old model tree, there is an obvious tendency that the error rate increases rapidly when the number of classes increases, as seen in Table 6.6. While in the new model, this error rate is maintained at a very low level, which indicates that it is possible to apply a tree classifier to recognize even more characters.

2) The recognition speeds are very high, considering the experiments were run in a high level language environment. The backtracking rate in Table 6.6 is the rate of samples which needs the fuzzy logic search. This figure is not high, because most samples were correctly recognized in the first phase of the search. Due to



I II III

I. Input character II. Result of the first search III. Result of the second search

Fig. 6.14 Mis-recognized Examples

this reason, even with the additional fuzzy logic search, the speed still remains very high.

3) The recognition rates by the fuzzy logic search for both trees with and without global training are very high. But it is obvious from the tables that the recognition rate has been improved further by global training, which justifies the new tree model. Besides, the new model gives a lot of flexibility, as analyzed in Chapter 5. The trade-off between time and speed, and that between error rate and rejection rate can be made in the new model, by adjusting some parameters when globally training the tree classifier (see Section 5.5). For example we can decrease the value of $\delta$ in global training (Algorithm 5.3.), or adjust $\lambda$ in eqn. 5.7, so that more i.d.'s can be collected in step 3) of Algorithm 5.3. In this way dissimilarity between "廷" and "垃" can be measured in the extended terminals after global training. Still we can adjust some threshold in the fuzzy-logic-search (Algorithm 5.4.) to find more candidate terminals, so that the above patterns can be re-recognized. But all these remedies need more time in the global training, as well as in the fuzzy-logic-search for all the patterns input to the tree. What we need in practice is a trade-off between efficiency and recognition rate.

The results in Sections 6.2 through 6.5 show the progress in the tree classifier research. The last few experiments on the new model tree show that the decision tree in a large character set has been greatly improved.

## 6.6. References

[1]. J. R. Ullman, "Advances in character recognition," in Applications of Pattern Recognition, K. S. Fu (ed.), pp.197-236, CRC Press Inc., 1982.

[2]. C. Y. Suen, Computational Analysis of Mandarin, Birkhauser Verlag, Basel, 1979.

[3]. P. A. Devijver, "Statistical pattern recognition," in Applications of Pattern Recognition, K. S .Fu ed., pp.15-36, CRC Press Inc., 1982.

[4]. T. M. Cover and Van Campenhout, "On the possible ordering in the measurement selection problem" IEEE Trans. Syst. Man. Cybern., SMC-7, 657, 1977.

[5]. P. M. Narendra and K. Fukunaga, " A branch and bound algorithm for feature subset selection," IEEE

Trans. Comput.,c-26, 917, 1977.

[6]. J. Kittler, "Feature set search algorithm," in Pattern Recognition and Signal Processing, C. H. Chen (ed.), pp. 41-60, Sijthoff Noordhoff, Northholand, 1978.

[7]. C. Y. Suen, "Feature extraction in automatic recognition of handprinted characters," Signal Processing, vol.4, 193-207, April 1982.

[8]. T. M. Cover, "The best two independent measurements are not the two best," IEEE Trans. Syst. Man and Cybern., SMC-4, pp. 116-117, 1974.

# CHAPTER SEVEN

# CONCLUSION

The decision tree has both prospect and difficulty in large character set recognition. The works covered in this thesis aim at overcoming the difficulties and make it possible to design a tree classifier with satisfactory performance in large character set recognition.

Based on entropy reduction, some analysis has been made, which is a new treatment and brings about some new results on tree classifiers. The (straight forward) search time and error rates have been shown both in the order O(H), and the memory requirement in the order O(H exp(H)), where H is Shannon's entropy measure of the given problem. These results reveal that the main difficulties in the tree approach to large character set problems are error rate accumulation and extra memory requirement due to overlap.

Under the guidance of these theoretical results, some design principles have been implemented, which include branch factor selection, entropy reduction, overlap control and the elimination of error accumulation effect.

A new clustering algorithm ISOETRP aiming at maximizing the Gain, defined as the ratio of entropy reduction over overlap, has been developed. With the help of the newly proposed overlap table technique, an interactive version of ISOETRP has been developed and achieved its objectives,

which has been found to be very effective and very powerful in tree classifier design. Overlap was considered an open problem in clustering. In ISOETRP, this problem has been solved interactively by the overlap table.

To solve the error accumulation problem, the branch-bound search with fuzzy membership function as the heuristic evaluation was proposed. In addition, the algorithm of tree classifier global training has been developed. The main characteristic of a tree classifier is that the final recognition of a pattern is derived from many local decisions, which has the advantage in time efficiency, but the disadvantage of having a "narrow view". After global training, the tree has many extended terminals where additional information can be stored to compensate for the localness problem in branch-bound search. In this way, the decision tree becomes a new model. The advantage of global training has a further benefit, i.e. the parameters can be adjusted easily, since it does not take much computing time.

Some related works on feature extraction and feature merit measurement have been included in this thesis. The phase feature was proposed and analyzed for recognition purpose, which enhanced the Fourier transforms in character recognition.

Many simulation experiments have been conducted to verify the above ideas, deductions and algorithms. The

number of character classes in these experiments ranges from 64 to 3200. The final results are very encouraging, which convince people that the decision tree approach has a great potential in large character set recognition.