Advances in Fuzzy Systems — Applications and Theory Vol. 10

FUZZY ALGORITHMS:

With Applications to Image Processing and Pattern Recognition

Zheru Chi
Hong Kong Polytechnic University, Hong Kong

Hong Yan
University of Sydney, Australia

Tuan Pham
University of Sydney, Australia

World Scientific
Singapore • New Jersey • London • Hong Kong

Published by

World Scientific Publishing Co. Pte. Ltd.

P O Box 128, Farrer Road, Singapore 912805

USA office: Suite 1B, 1060 Main Street, River Edge, NJ 07661

UK office: 57 Shelton Street, Covent Garden, London WC2H 9HE

Library of Congress Cataloging-in-Publication Data
Chi, Zheru.
Fuzzy algorithms : with applications to image processing and pattern recognition / Zheru Chi, Hong Yan, Tuan Pham.
p. cm.
Includes bibliographical references and index.
ISBN 9810226977
1. Image processing -- Digital techniques. 2. Pattern recognition systems. 3. Fuzzy algorithms. I. Yan, Hong. II. Pham, Tuan. III. Title.
TA1637.C48 1996
006.4'2'015113--dc20 96-31969
CIP

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library.

This book is printed on acid-free paper.

Printed in Singapore by Uto-Print

Preface

In the fields of image processing and pattern recognition, a great number of different techniques have been investigated, developed, and implemented in the last few decades. Starting from statistical models to neural network algorithms, researchers from all over the world have made a good number of contributions to the development of improved methodologies for solving difficult problems in these fields.

Sine L. A. Zadeh introduced the fuzzy set theory in 1965, the fuzzy system approach has been an increasing interest of many scientists and engineers for opening up a new area of research and problem solving. Many books on fuzzy systems have since been published. However, most of them are devoted to the theoretical developments and their applications in control systems. In the last few years, we have also seen the booming of applications of fuzzy algorithms in image processing and pattern recognition. The interest in using fuzzy algorithms comes from the facts that: (1) fuzzy rules are found to be naturally effective for any human-like cognition systems such as image understanding and pattern recognition, and (2) the theory of fuzzy sets provides a good platform for dealing with noisy, and imprecise information which is often encountered in our daily life.

This book does not focus on the theoretical research of fuzzy algorithms, but it rather provides a comprehensive discussion on several issues of application-oriented methodologies using fuzzy algorithms for image processing and pattern recognition. These include the segmentation of a variety of images and characterization of tissues in magnetic resonance (MR) images, parameter identification for mining deposits, and printing and handwritten character recognition.

This book is mainly aimed at three groups of readers, (1) those who are familiar with fuzzy algorithms and want to identify the useful applications of fuzzy logic, (2) those who have been articulating in the fields of image processing and pattern recognition for some years and want to find some new and interesting topics to work on, and (3) those who want to know more about both fuzzy algorithms and their practical applications. As an application-oriented book, university students, researchers, and engineers would find the book useful in many practical aspects. In addition, this book can serve as supplementary reading material for university courses in areas of image processing, pattern recognition, artificial intelligence, and fuzzy systems.

This book is divided into seven chapters. Following an introductory chapter is the generation of membership functions from training examples. Then each of the remaining chapters is devoted to a specific fuzzy algorithm or a class of fuzzy algorithms, and their applications in image processing and pattern recognition.

In **Chapter 1** we introduce the basic concepts and ideas of fuzzy set theory, including probability and fuzziness, basic properties, some operations on fuzzy sets, and fuzzy relations. The chapter ends with a brief discussion on fuzzy models for image processing and pattern recognition.

In **Chapter 2** three clustering-based techniques (c-means, adaptive vector quantization, and the self-organizing map) for membership function generation will be presented. Two techniques for tuning membership functions using the gradient descent algorithm and a neural network approach are also discussed.

In **Chapter 3** we discuss Huang and Wang's fuzzy thresholding algorithm following the introduction of the threshold selection method based on statistical decision theory and non-fuzzy thresholding algorithms. A unified description of all fuzzy or non-fuzzy thresholding methods is then given and the extension of the methods to multilevel thresholding is proposed. Finally, the applications of these thresholding methods to real images are provided.

In **Chapter 4** we describe and compare the hard and fuzzy c-means algorithms, which can be used for clustering when the number of clusters is known. We then describe three cluster validity measures which can be used to analyze the quality of fuzzy clustering procedures. The chapter ends with a section on applying the fuzzy clustering algorithms to several real world image processing and pattern recognition problems.

In **Chapter 5** we first describe how to use membership functions to measure similarities between line segments. A basic algorithm for line pattern matching based on spatial relations of the lines in a prototype and the input pattern is discussed. We then propose more sophisticated algorithms to deal with noisy patterns and geometrically distorted patterns. To demonstrate the usefulness of the proposed line pattern matching algorithms, applications to Chinese character recognition and point pattern matching are illustrated.

In **Chapter 6** we present three fuzzy rule generation techniques based on learning from examples: Wang and Mendel's method, ID3 decision rules, and Krishnapuram's neural network approach. We then describe the minimization of fuzzy rules based on Karnaugh maps proposed by Hung and Fernandez and discuss various defuzzification methods for fuzzy rule-based recognition systems. The chapter ends with a section on applications of fuzzy rule recognition systems for segmentation of geographic map images and recognition of printed upper-case English letters and handwritten digits.

In **Chapter 7** we introduce several multi-classifier combination techniques for improving classification performance including voting schemes, a maximum posteriori probability based method, a multilayer perceptron approach, and a fuzzy integral model. The chapter ends with a section on applying these combination classifiers to handwritten numeral character recognition.

Acknowledgements

Much of our work described in this book was funded by the Australian Research Council (ARC), the Australian Department of Industry, Science and Technology (DIST), the Japanese Ministry of Industry and International Trade (MITI) and the Real World Computing Partnership (RWCP). We would like to thank these funding bodies for their support.

Some work reported in the book was done by our colleagues, Ms. Jing Wu and Dr. Mark Suters. We would like to express our sincere appreciation for their contribution. We are also deeply grateful to our publisher, World Scientific Publishing Co., for their helpful suggestions in the production of this book.

Finally, we would like to offer our deepest and most heartfelt thanks to our wives and children for their support, encouragement and patience.

Contents

Preface v

1 Introduction 1
1.1 Basic Concepts of Fuzzy Sets . . . 1
1.1.1 Probability and Fuzziness . . . 1
1.1.2 Fuzzy Sets . . . 3
1.1.3 Properties of Fuzzy Sets . . . 5
1.1.4 Operations on Fuzzy Sets . . . 8
1.2 Fuzzy Relations . . . 11
1.3 Fuzzy Models for Image Processing and Pattern Recognition . . . 14

2 Membership Functions 17
2.1 Introduction . . . 17
2.2 Heuristic Selections . . . 18
2.3 Clustering Approaches . . . 23
2.3.1 C-means Clustering Approach . . . 24
2.3.2 Learning with Adaptive Vector Quantization . . . 26
2.3.3 Self-organizing Map Approach . . . 28
2.3.4 Merging Neighboring Membership Functions . . . 33
2.4 Tuning of Membership Functions . . . 35
2.4.1 Gradient Descent Algorithm . . . 35
2.4.2 Neural Network Approach . . . 38
2.5 Concluding Remarks . . . 44

3 Optimal Image Thresholding 45
3.1 Introduction . . . 45
3.2 Threshold Selection Based on Statistical Decision Theory . . . 47
3.2.1 Statistical Decision Rule . . . 47
3.2.2 Gaussian Distributions . . . 50
3.3 Non-fuzzy Thresholding Algorithms . . . 52
3.3.1 Model Fitting Method . . . 52
3.3.2 Otsu's Thresholding Method . . . 54
3.3.3 Minimum Error Thresholding Method . . . 56

3.4 Fuzzy Thresholding Algorithm 57
3.4.1 Fuzzy Membership Functions 57
3.4.2 Threshold Selection Based on Fuzzy Measure 58
3.4.3 Fast Computation 60
3.5 Unified Formulation of Three Thresholding Algorithms 61
3.5.1 Analytical Formulation 61
3.5.2 Analysis of Cost Functions 62
3.6 Multilevel Thresholding 63
3.6.1 Generalization of Model Fitting Method 66
3.6.2 Generalization of Non-iterative Methods 68
3.7 Applications 69
3.7.1 Segmentation of Document Images 69
3.7.2 Segmentation of Building Images 70
3.7.3 Segmentation of Human Images 71
3.8 Concluding Remarks 84

4 Fuzzy Clustering 85
4.1 Introduction 85
4.2 C-means Algorithm 86
4.3 Fuzzy C-means Algorithm 87
4.4 Comparison between Hard and Fuzzy Clustering Algorithms 89
4.5 Cluster Validity 91
4.6 Applications 93
4.6.1 Characterization of Tissues in MR Images 93
4.6.2 Segmentation of Natural Color Images 95
4.6.3 Fuzzy Parameter Identification 100
4.7 Concluding Remarks 106

5 Line Pattern Matching 109
5.1 Introduction 109
5.2 Similarity Measures between Line Segments 112
5.2.1 Location Similarity 112
5.2.2 Orientation Similarity 113
5.2.3 Relation Similarity 114
5.3 Basic Matching Algorithm 117
5.3.1 Entropy Measure 117
5.3.2 Matching Costs 119
5.3.3 Examples 121
5.4 Dealing with Noisy Patterns 125
5.4.1 Detecting and Removing Extra Lines 125
5.4.2 Detecting and Recovering Missing Lines 127
5.4.3 Summary of Matching Algorithms 129
5.5 Dealing with Rotated Patterns 129

5.5.1 Basic Algorithm 129
5.5.2 Fast Algorithm 131
5.6 Applications 134
5.6.1 On-line Handwritten Chinese Character Recognition 134
5.6.2 Point Pattern Matching 134
5.7 Concluding Remarks 135

6 Fuzzy Rule-based Systems 139
6.1 Introduction 139
6.2 Learning from Examples 141
6.3 Decision Tree Approach 145
6.3.1 Decision Trees and Rules 145
6.3.2 Simplified Decision Trees 150
6.3.3 Fuzzified Decision Rules 150
6.4 Fuzzy Aggregation Network Approach 152
6.5 Minimization of Fuzzy Rules 153
6.5.1 Minimization Based on Data Clustering 156
6.5.2 Minimization Based on Karnaugh Maps 156
6.6 Defuzzification and Optimization 159
6.6.1 Maximum Matching 160
6.6.2 Maximum Accumulated Matching 160
6.6.3 Centroid Defuzzification 161
6.6.4 Two-layer Perceptron Approach 161
6.7 Applications 163
6.7.1 Segmentation of Map Images 163
6.7.2 Printed Upper-case English Letter Recognition 174
6.7.3 Handwritten Numeral Recognition 178
6.8 Concluding Remarks 186

7 Combined Classifiers 189
7.1 Introduction 189
7.2 Voting Schemes 190
7.3 Maximum Posteriori Probability 191
7.4 Multilayer Perceptron Approach 192
7.5 Fuzzy Measures and Fuzzy Integrals 192
7.5.1 Fuzzy Measures 194
7.5.2 Fuzzy Integrals 195
7.5.3 A Fuzzy Integral Model for Classifier Fusion 195
7.6 Applications 197
7.6.1 Optimized Nearest Neighbor Classifier 198
7.6.2 Markov Chain-based Classifier 201
7.6.3 Combined Classifiers 204
7.7 Concluding Remarks 211

BIBLIOGRAPHY **213**

INDEX **221**

Chapter 1

Introduction

1.1 Basic Concepts of Fuzzy Sets

1.1.1 Probability and Fuzziness

The notion of probability has apparently had a pervasive influence upon the computational frameworks in science and engineering. In the terminology of set theory and the mathematical theory of probability, an event $E \in \sigma$-field is defined as a subset of the sample space Ω which is a collection of all possibilities or sample points in a probabilistic problem. A probability measure P over a measurable space (Ω, σ) is a real function which assigns a normed measure to an event E, denoted as $P(E)$, such that

$$P(E) \geq 0, \ \forall E \in \sigma$$

and

$$P(\Omega) = 1$$

If $\{A_i\}$ is any collection of disjoint events, then its probability measure is additive, that is

$$P\left(\cup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} P(A_i)$$

From the above definition, an event E in terms of probability theory is always referred to as a subset of a particular sample space Ω. Therefore it consists of one or more sample points, and the realization of these sample points indicates its occurrence. However, many real-world events that we encounter daily are perceived to be vague or ill-defined rather than being a probabilistic problem. For example, when solving a real-life problem, one agrees that the rule of "*either ... or* " should be replaced by that of "*both ... and* ". In image processing, the grey level of an image element is defined as *both* black *and* white to a certain degree of the two attributes respectively, where the fullness of being black or white is only a special case. This natural concept of vagueness was not formalized in the scientific community until the work by Zadeh in 1965 [1]. The concept of fuzzy sets is considered being a generalization of the notion of ordinary sets. The underlying logic of the fuzzy-set theory is that it allows an event to belong to more than one sample space where sharp boundaries between spaces are hardly found. As another example, when we are asked whether it will rain tomorrow, we would normally give an answer to the likelihood of raining with a degree of confidence unless we are totally sure about the occurrence of the event. This is a case where we can express our prediction using the numerical scale of probability. If we say that the chance of rain tomorrow is 40%, then there is an implication that the chance of no rain is 60%. However, in daily living it is unusual for us to give a crisp estimate for such an event. We rather include some linguistic expressions such as "It is *very likely*" to rain, and "the chance of rain is *about* 40%", *etc.* It can be perceived that the terms in *italics* are involved in the fuzziness of our probabilistic problem as our estimates of rain are not totally dependent upon the deterministic and/or probabilistic factors but also our *subjective feeling* about the phenomena.

Again, as in the area of image analysis, it is more natural to treat transient regions between two areas in an image as fuzzy domains in which the degree of *fairness* and the pixels having *almost the same* color and the *gradual* change in color are in fact the expressions of fuzziness. For pattern recognition techniques which may be best defined as the "methods that search for structures in data" [2], scientists are interested in the distribution of data. These data may represent the structures of certain patterns needed to be identified. Conventional patterns of data are classified into three categories: regular, random, and clustered. There are, of course, hybrid types of these main patterns. But in circumstances where the amount of data points is not sufficient to support a probability model, then this kind of pattern yields itself to a fuzzy one.

In summary, fuzzy-set algorithms outperform other conventional methods when they are appropriately applied to solve the problems of fuzzy uncertainty. Some main useful features of the fuzzy-set methodologies can be outlined as follows:

1. Fuzzy logic provides a systematic basis for quantifying uncertainty due to vagueness and incompleteness of information.

2. Classes with unsharp boundaries can be easily modeled using fuzzy sets.

3. Fuzzy reasoning is a formalism that allows the use of expert knowledge, and is able to process this expertise in a structured and consistent way.

4. There is no broad assumption of complete independence of the evidence to be combined using fuzzy logic, as required for other subjective probabilistic approaches.

5. When the information is inadequate to support a random definition, the use of

probabilistic methods may be difficult. In such cases, the methods of fuzzy sets are promising.

1.1.2 Fuzzy Sets

The theory of fuzzy sets was first introduced by Zadeh [1]. The concept of fuzzy sets emerged when it was realized that it may not be possible to model ill-defined systems with precise mathematical assumptions of the classical methods, such as probability theory. Although it is only about three decades since the present concept of fuzzy sets emerged, it dates back to the work of a Polish mathematician, Lukasiewiecz, on multi-valued logic. However, multi-valued logic has not been used significantly in logical systems because of its restricted framework. Gaines and Kohout [3] pointed out that although fuzzy set theory is distinctive from probability theory, there are still certain relationships between the two theories in formal and practical aspects. One of the most upsetting concepts which was raised by the philosophy of fuzzy logic is that the Law of Excluded Middle, which was defined by Aristotle, is no longer completely true. The Law of the Excluded Middle states "X must be either Z or not Z". In other words, a person is either tall or not tall. However, in terms of fuzzy logic, a person can be both tall and not tall in which the difference is the degrees of certainty assigned to the fuzzy sets of "tall" and its complement "not tall". Thus, the intersection of a fuzzy set and its complement is not always an empty set, and contradictory objects can be members of the same fuzzy set.

Figures 1.1 and 1.2 show the different concepts of describing "true" and "false" by crisp sets and fuzzy sets respectively. What really matters for a fuzzy set is the degree of truth assigned for each of its members. This perception of vagueness turns out to be natural and more appealing to our daily concepts. Fuzzy set theory is not a theory that permits vagueness in our computations, but it is rather a methodology to show how to tackle uncertainty, and to handle imprecise information in a complex situation.

Let X be a collection of objects or a universe of discourse, then a fuzzy set A in X is a set of ordered pairs

$$A = \{\mu_A(x)/x\} \tag{1.1}$$

where $\mu_A(x)$ is the characteristic function (or the membership function) of x in A. The slash "/" is used simply as a separator between the real value x and its membership grade $\mu_A(x)$, and $\mu_A(x)$ may take any real values in the interval [0,1], $\mu_A : X \to [0,1]$.

If A contains only 0 and 1, then A is non-fuzzy and its membership function becomes identical with the characteristic function of a crisp set.

If the membership function of A is discrete, then A is written as

$$A = \mu_1/x_1 + \mu_2/x_2 + ... + \mu_n/x_n = \sum_{i=1}^{n} \mu_i/x_i \tag{1.2}$$

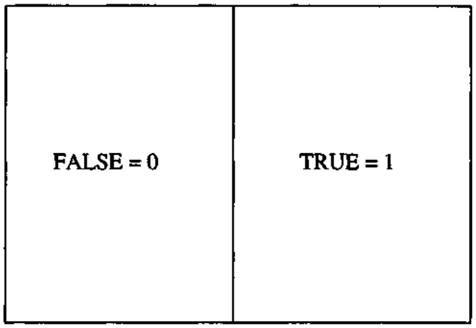


Figure 1.1 Crisp sets of TRUE and FALSE.

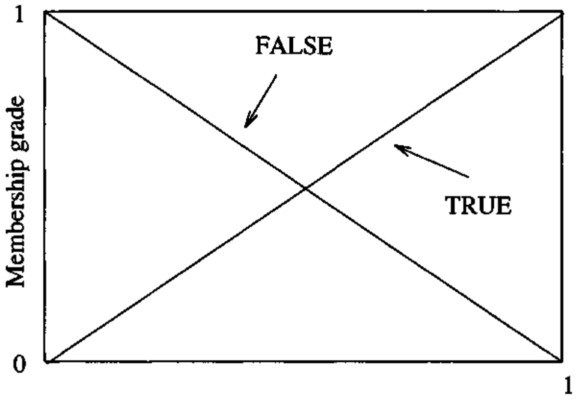


Figure 1.2 Fuzzy sets of TRUE and FALSE.

where "+" and the "$\sum$" stand for the union.

When the membership function is a continuum, the fuzzy set A is written as

$$A = \int_X \mu_A(x)/x \tag{1.3}$$

where the integral denotes the union of the fuzzy singletons $\mu_A(x)/x$.

1.1.3 Properties of Fuzzy Sets

- *Normality*: A fuzzy set is said to be normal if the greatest value of its membership function is unity:

  $$\vee_x \, \mu(x) = 1 \tag{1.4}$$

  where $\vee_x$ stands for the supremum of $\mu(x)$ (the least upper bound).

  Otherwise the fuzzy set is subnormal.

  Figures 1.3(a) and 1.3(b) show normal and subnormal fuzzy sets.

- *Convexity*: A fuzzy set A is convex if and only if the sets A_α defined as

  $$A_\alpha = \{x/\mu_A(x) \; \geq \alpha\} \tag{1.5}$$

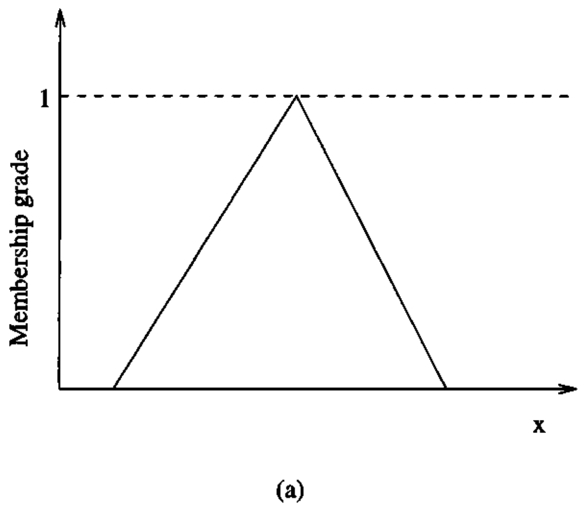
  are convex for all $\alpha \in [0,1]$. Figures 1.4(a) and 1.4(b) show the convex and non-convex fuzzy sets respectively. A_α is the level-cut set whose membership function is equal or greater than α.

- *Crossover point*: A crossover point of a fuzzy set A is an element whose membership grade in A is equal to 0.5.

- *Fuzzy singleton*: A fuzzy singleton is a fuzzy set which only has a membership grade for a single value. Let A be a fuzzy singleton of a universe of discourse X, $x \in X$, then A is written as

  $$A = \mu/x. \tag{1.6}$$

  With the above definition, a fuzzy set can be considered as the union of fuzzy singletons.



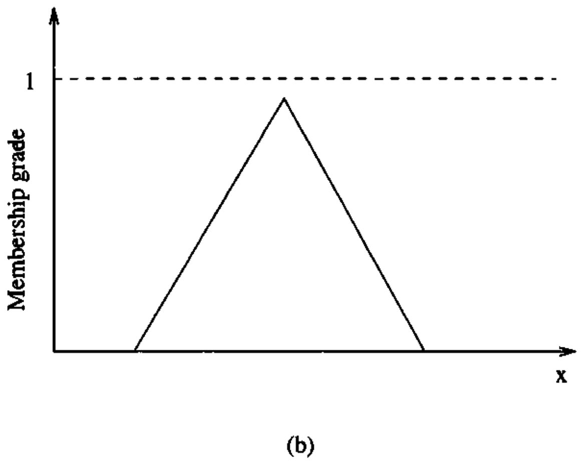


Figure 1.3 (a) A normal fuzzy set; (b) a subnormal fuzzy set.

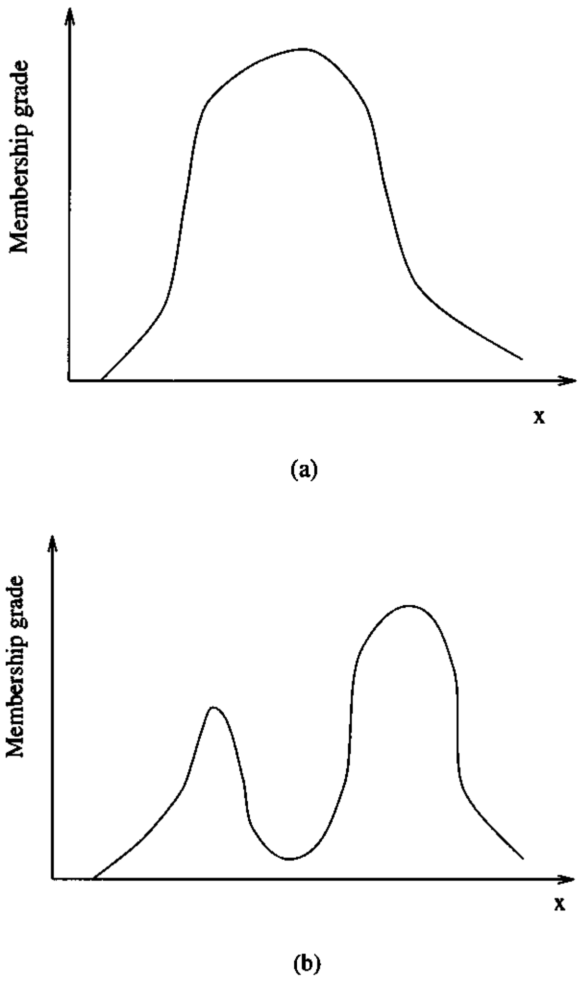


Figure 1.4 (a) A convex fuzzy set; (b) a non-convex fuzzy set.

1.1.4 Operations on Fuzzy Sets

The operations on fuzzy sets presented in this section are based on the works of Zadeh and they should not be considered as a complete collection. The intention is to provide adequately the mathematical aspects of fuzzy set theory in relation to discussions and developments in the following chapters. The original information of fuzzy set theory can be found from the excellent collection of Zadeh's papers edited by Yager *et al* [4]. Other excellent books on fuzzy sets include [5], [6], [7].

- *Fuzzy intersection*: Figure 1.5 shows the intersection of two fuzzy sets A and B. The fuzzy intersection is interpreted as "A AND B", which takes the minimum value of the two membership grades.

  $$A(x) \cap B(x) = \sum \mu_A(x) \wedge \mu_B(x) \tag{1.7}$$

  where $\wedge$ denotes the minimum operator.

- *Fuzzy union*: Figure 1.6 shows the union of two fuzzy sets A and B. The fuzzy union is interpreted as "A OR B", which takes the maximum value of the two membership grades.

  $$A(x) \cup B(x) = \sum \mu_A(x) \vee \mu_B(x) \tag{1.8}$$

  where $\vee$ denotes the maximum operator.

- *Fuzzy complement*: The complement of a fuzzy set A, which is understood as "NOT (A)", is defined by (see Fig. 1.7)

  $$\bar{A} = \sum 1 - \mu_A(x) \tag{1.9}$$

  where $\bar{A}$ stands for the complement of A.

- *Convex combination*: The convex combination is an operator which combines different fuzzy sets into a single fuzzy set using the weights assigned to each fuzzy set. The total membership function $\mu_T(x)$ as a result of the convex combination of the membership functions $\mu_{A_1}, ..., \mu_{A_n}$ is defined by

  $$\mu_T(x) = w_1(x)\mu_{A_1}(x) + w_2(x)\mu_{A_2}(x) + ... + w_n(x)\mu_{A_n}(x) \tag{1.10}$$

  where $w_1, w_2, ..., w_n$ are the weights for the fuzzy sets $A_1, A_2, ..., A_n$, respectively such that:

  $$w_1(x) + w_2(x) + ... + w_n(x) = 1 \tag{1.11}$$

  with the understanding that the "+" sign in Eqs. (1.10) and (1.11) denotes an arithmetic addition.

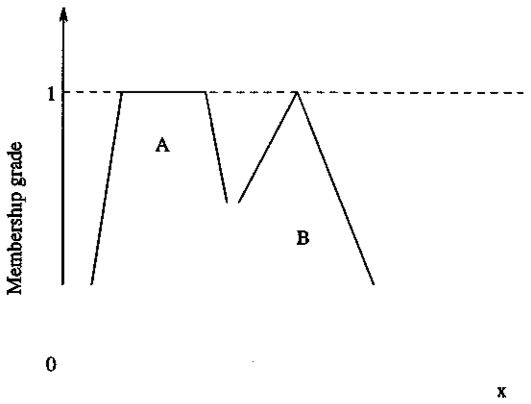


Figure 1.5 Intersection of fuzzy sets A and B (shaded area).

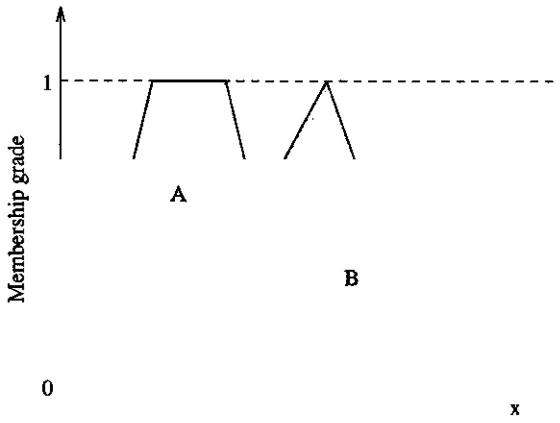


Figure 1.6 Union of fuzzy sets A and B (shaded area)

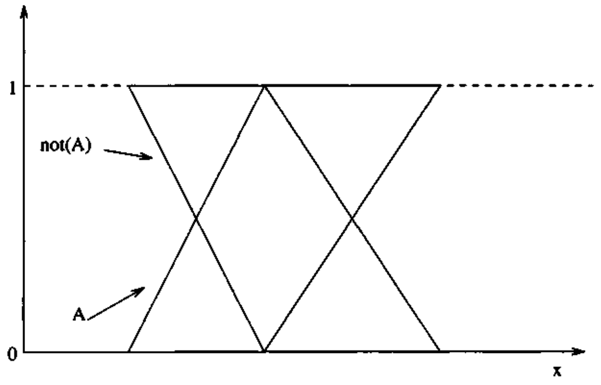


Figure 1.7 Fuzzy set A and its complement.

- *Fuzzy concentration*: The concentration of a fuzzy set produces a reduction by squaring the membership function of that fuzzy set. If a fuzzy set A is written as

$$A = \{\mu_1/x_1 + \mu_2/x_2 + ... + \mu_n/x_n\}$$

  then the fuzzy concentrator applied to a fuzzy set A is defined:

$$CON(A) = A^2 = \{\mu_1^2/x_1 + \mu_2^2/x_2 + ... + \mu_n^2/x_n\} \tag{1.12}$$

  where CON(A) denotes the concentrator applied to A.

- *Fuzzy dilation*: The fuzzy dilation is an operator which increases the degree of belief in each object of a fuzzy set by taking the square root of the membership function. Clearly, the operation of dilation has an opposite effect to that of concentration defined in Eq. (1.12):

$$DIL(A) = A^{0.5} = \{\mu_1^{0.5}/x_1 + \mu_2^{0.5}/x_2 + ... + \mu_n^{0.5}/x_n\} \tag{1.13}$$

  where DIL(A) denotes the dilator applied to the fuzzy set A.

- *Fuzzy plus and fuzzy minus*: The operations of plus and minus applied to a fuzzy set give an intermediate effect of concentration and dilation respectively. If A is a fuzzy set, then plus and minus may be defined for example as

$$\mathrm{Plus}(A) = A^{1.25} \tag{1.14}$$

$$\mathrm{Minus}(A) = A^{0.75} \tag{1.15}$$

- *Fuzzy intensification:* Intensification is an operator which increases the membership function of a fuzzy set above the crossover point (at $\alpha = 0.5$), and decreases that of a fuzzy set below the crossover point.

  If A is a fuzzy set, $x \in A$, then the intensification applied to A is defined as

$$\begin{aligned} \mu_{INT(A)}(x) &\geq \mu_A(x),\ \mu_A(x) \geq 0.5 \\ \mu_{INT(A)}(x) &\leq \mu_A(x),\ \mu_A(x) \leq 0.5 \end{aligned} \tag{1.16}$$

  A typical expression for the application of the intensifying operator is the S-function proposed by Zadeh [8] defined as

$$S(x) = \begin{cases} 0 & \text{if } \ x < 0 \\ 2x^2 & \text{if } \ 0 \leq x \leq 0.5 \\ 1 - 2(1-x)^2 & \text{if } \ 0.5 \leq x \leq 1 \\ 0 & \text{if } \ x > 1. \end{cases} \tag{1.17}$$

- *Bounded sum:* The bounded sum of two fuzzy sets A and B in the universes X and Y with the membership functions $\mu_A(x)$ and $\mu_B(y)$ respectively, is defined by

$$A \oplus B = \mu_{A \oplus B} = 1 \wedge (\mu_A(x) + \mu_B(y)) \tag{1.18}$$

  where the sign "+" is the arithmetic operator.

- *Bounded product:* The bounded product of two fuzzy sets A and B as described in the bounded sum is defined by

$$A \otimes B = \mu_{A \otimes B} = 0 \vee (\mu_A(x) + \mu_B(y) - 1). \tag{1.19}$$

1.2 Fuzzy Relations

Fuzzy relations have been extensively investigated in literature due to their inherently important applications. The use of fuzzy relations provides a natural way to infer a conclusion from a human expression which includes vagueness. In terms of mathematical formality, a fuzzy relation can be defined as follows:

A fuzzy relation R of a fuzzy set A to a fuzzy set B is a fuzzy subset of the Cartesian product $X \times Y$ which is the collection of ordered pairs (x, y), $x \in X$, $y \in Y$. The following expression is a fuzzy binary relation where R is characterized by the bivariate membership function $\mu(x, y)$ of the associated pair (x, y):

$$R \trianglerighteq \int_{X\times Y} \mu_R(x,y)/(x,y) \tag{1.20}$$

where the symbol $\trianglerighteq$ stands for "defined as".

For an n-ary fuzzy relation , R is written as:

$$R \trianglerighteq \int_{X_1\times X_2\times \ldots \times X_n} \mu_R(x_1,x_2,\ldots,x_n)/(x_1,x_2,\ldots,x_n), \;\; x_i \in X_i, \; i = 1,n. \tag{1.21}$$

The composition of the two binary fuzzy relations $R_1(X, Y)$ and $R_2(Y, Z)$ yields another fuzzy relation $R_3(X, Z)$ which is defined by

$$\begin{aligned} R_3(X,Z) &= R_1(X,Y) \circ R_2(Y,Z) \\ &= \int_{X\times Z} \vee \left(\mu_{R_1}(x,y) \wedge \mu_{R_2}(y,z)\right)/(x,z) \end{aligned} \tag{1.22}$$

where "o" denotes the operation of fuzzy composition.

Based on the concept of fuzzy relation, some rules for fuzzy inference are also proposed by Zadeh [9], [10]. These are known as:

- *Entailment rule.* The entailment principle allows us to infer from a fuzzy proposition "X is A" to a less specific proposition "X is B". The entailment rule is expressed in a canonical form by

$$\begin{array}{l} \text{X is A} \\ \underline{A \subset B \rightarrow \mu_A(x) \leq \mu_B(x),\; x \in X} \\ \text{(therefore) X is B.} \end{array} \tag{1.23}$$

  It should be noted that the straight line is used to separate the premises from the conclusion.

- *Conjunctive rule.* If the fuzzy premises are expressed in the form "X is A_1, X is A_2, ..., X is A_n.", then the conclusion induced by the conjunctive rule is the intersection of all fuzzy premises - all premises are connected by the operator AND ($\wedge$).

$$\begin{array}{c} \text{X is } A_1 \\ X \text{ is } A_2 \\ \dots \\ X \text{ is } A_n \\ \hline \text{(therefore) } X \text{ is } A_1 \cup A_2 \cup \dots \cup A_n \rightarrow \mu_{A_1 \cup A_2 \cup \dots \cup A_n}(x) = \\ \mu_{A_1}(x) \wedge \mu_{A_2}(x) \wedge \dots \wedge \mu_{A_2}(x) \end{array} \tag{1.24}$$

- *Projection rule.* The projection of the binary relation R of X and Y on the domain of X is defined as

$$\begin{array}{c} (X,Y) \text{ is } R \\ \hline \text{(therefore) } X \text{ is } XR \end{array} \tag{1.25}$$

where XR is the projection of R on X, and the membership function is given by

$$\mu_{XR}(x) = \vee_y \mu_R(x,y), \quad x \in X,\ y \in Y$$

- *Compositional rule.* The compositional rule of inference can be applied to infer a new fuzzy subset $B'(Y)$ by $A'(X) \circ R_{A\times B}(X,Y)$ in which the membership function of B' is constructed by the max-min product of the row vector of $\mu_{A'}(x)$ and the relation matrix of $\mu_{A\times B}(x,y)$, for $x \in X$, and $y \in Y$, that is,

$$\begin{array}{l} \text{A is X} \\ \text{B is Y} \\ \text{(X,Y) is R} \\ A' \text{ is } X' \\ \hline \text{(therefore)} \quad B' \text{ is } A' \circ R. \end{array} \tag{1.26}$$

- *Generalized modus ponens.* The term modus ponens means "method of affirming" since the conclusion is an affirmation. The conclusion which is inferred by the generalized modus ponens is defined by

$$\begin{array}{ll} \text{(Major premise:)} & \text{If X is B then Y is C} \\ \text{(Minor premise:)} & \text{X is A} \\ \hline \text{(Conclusion:)} & \text{Y is A } \circ (\bar{B} \oplus C) \end{array} \tag{1.27}$$

where $\bar{B}$ is the negation of B.

It is noted that the generalized modus ponens does not necessarily require the antecedent "X is B" in the major premise to be identical with the minor premise "X is A". This property of free identity is not valid for the classical modus ponens in the binary logical argument.

1.3 Fuzzy Models for Image Processing and Pattern Recognition

The theory of fuzzy sets has immediately found its potential application to the fields of pattern recognition and image processing. Recently a monograph of collected published papers on fuzzy models for pattern recognition has been published by Bezdek and Pal [2] to present several significant contributions starting from concepts to applications of fuzzy models for the solutions of real-life problems.

There are several fuzzy-set models for solving problems in pattern recognition and image processing. Some popular models are: the use of fuzzy membership function, fuzzy clustering, fuzzy-rule based systems, fuzzy entropy (measure of fuzziness), fuzzy measure and fuzzy integral. These mathematical tools for fuzzy-set modeling shall be briefly discussed as follows.

1. **Fuzzy membership functions**: The membership function is the underlying power of every fuzzy model as it is capable of modeling the gradual transition from a less distinct region to another in a subtle way. This fundamental framework has been applied to measure the fuzziness inherently existing in the recognition of handwritten numeral characters such as the definition of "*straightness*" and "*orientation*" for feature classification. The direct use of fuzzy membership functions is also found effective for image enhancement where different types of membership functions are used to reduce the amount of iterations carried out by a relaxation technique and provide a better way to handle the uncertainty of the image histogram (Chapter 3).

2. **Fuzzy clustering**: Clustering method has been a dominant solution method for pattern classification problems. However, in nature there exist only a few cases where the differences between the patterns are clear-cut. Therefore, the implementation of fuzzy-set mathematics in conventional clustering algorithms is naturally essential. Many problems involving classification in image analysis and computer vision have been effectively solved using fuzzy clustering techniques (Chapter 4).

3. **Fuzzy pattern matching:** Many pattern recognition problems can be simplified as a point or line pattern matching task. Fuzzy algorithms, which can deal with ambiguous or fuzzy features of noisy point or line patterns, have been found to be particularly useful in line pattern matching. Many algorithms, including various fuzzy relaxation based techniques [11], [12], have been developed and applied to a number of pattern recognition applications (Chapter 5).

4. **Fuzzy rule-based systems**: The rule-based system is one of the most active research areas in artificial intelligence. It is actually a dominated platform for various expert systems. We believe that we human beings take a similar approach to perceive the world around us in a very robust way. In the real

world, almost everything is uncertain, therefore, a fuzzy rule-based system is expected to achieve a better performance than a crisp rule-based system in dealing with ambiguous, uncertain and fuzzy data. Fuzzy rule systems have found applications in many fields including control, decision making, and pattern recognition [2], [13], [14], [15]. A fuzzy rule base consists of a set of fuzzy IF-THEN rules which together with an inference engine, a fuzzifier, and a defuzzifier form a fuzzy rule-based system. The role of a fuzzifier is to map crisp points in the input space to fuzzy subsets by applying membership functions to inputs. System performance is very much dependent of chosen membership functions, in particular, for the systems dedicated for image processing and pattern recognition. In Chapter 2, we shall discuss how to generate a set of membership functions which can reflect the actual data distribution by using the training data. Also discussed in this chapter are two approaches to tune the membership functions for an improved system performance. Fuzzy rules can either be extracted from expert knowledge or learned from numerical data. In Chapter 6, three techniques to produce fuzzy rules from the training data are presented. As the minimization of a fuzzy rule set is the constant motivation technically and economically, we devote a section in this chapter to deal with this problem. Also discussed in Chapter 6 are various defuzzification methods for fuzzy pattern recognition models and optimized defuzzification by using feedforward neural networks.

5. **Fuzzy entropy**: The concept of fuzzy entropy or measure of fuzziness was first introduced by De Luca and Termini [16]. The entropy of a fuzzy set is a functional to measure the degree of fuzziness of a fuzzy set, which is based on Shannon's function. This fuzzy-entropy model has been applied to provide a quantitative measure of ambiguity to the problems of grey-tone image enhancement [17]. The measure of fuzziness based on metric distance was also found useful in medical diagnosis [18], [19] as the symptoms of several diseases expressed by patients overlapped each other.

6. **Fuzzy measure and fuzzy integral**: Based on the notion of fuzzy set theory, fuzzy measure and fuzzy integral were introduced by Sugeno [20]. Fuzzy measure provides a non-additive measure of multi-attributes, and fuzzy integral is a kind of information-aggregation operator, which integrates an information function over the fuzzy measures of all the subsets of the attributes in a nonlinear procedure. The application of fuzzy measure and fuzzy integral to image processing and computer vision have been investigated actively in recent years. Some of these investigations include the use of fuzzy integral for multi-classifier [21] (Chapter 7), image segmentation and image understanding as reported in [22], and character recognition (Chapter 7). The potential of this fuzzy model to pattern recognition and image processing problems is certain.

Chapter 2

Membership Functions

2.1 Introduction

Choosing membership functions, the first and an essential step for fuzzy logic applications, is more important for a fuzzy pattern recognition model than for a fuzzy control system. This is because the former is an open loop system, unlike a closed loop system for the latter, and the membership functions have a much greater impact on the system performance. Besides the heuristic selection of membership functions, as is usually done in developing a fuzzy control system, many other techniques have been proposed to produce membership functions which reflect the actual data distribution by using unsupervised or supervised learning algorithms. Learning membership functions are not only important but also feasible for image pattern recognition because of a large number of training patterns available.

Whether a statistical approach or a fuzzy system should be used to solve a problem with uncertain and/or noisy data has been debated for a quite long period. According to the original intention in introducing fuzzy systems, membership functions should be something subjective which clearly differs from objective probabilities. However, a set of subjective membership functions either are too difficult to choose due to the lack of understanding of the human approach on the problem or cannot produce a satisfactory result. Therefore, many researchers have proposed the combined statistical method and fuzzy system for problem solving. Using membership functions which are generated from training data by one of various clustering techniques is one way to achieve this combination.

A clustering algorithm can be applied to estimate the actual data distribution and the resulting clusters can be used to produce the membership functions which will interpret the data better. Dickerson and Kosko have proposed a learning technique for constructing membership functions by the adaptive vector quantization (AVQ) [23]. Besides the AVQ learning technique, we have worked on using c-means clustering and the self-organizing map for producing membership functions which reflect the actual data distribution in the input space [24], [25], [26], [27]. To improve fuzzy system performance, the chosen or generated membership functions can be further tuned by using the gradient descent algorithm [28]. Neural networks, as an effective optimization approach, can also be used for tuning membership functions [29].

Following this instruction, we first show some commonly adopted membership functions and give a few definitions on membership functions in Section 2.2. Then in Section 2.3, three clustering techniques: c-means, adaptive vector quantization, and the self-organizing map for producing membership functions are discussed. A merging scheme which combines neighboring fuzzy sets is also proposed in this section. Tuning of membership functions by using the gradient descent algorithm and a neural network is described in Section 2.4.

2.2 Heuristic Selections

The choice of membership functions is usually problem dependent. This is often determined heuristically and subjectively [2], [28]. Triangular, trapezoidal, and bell-shaped functions are three commonly used membership functions (see Fig. 2.1), which are denoted as $\Lambda(x;a,b,c)$, $\Pi(x;a,b,c,d)$, and $\pi(x;b,c)$, respectively. They are defined as

$$\Lambda(x;a,b,c) = \begin{cases} 0 & x \le a, \\ (x-a)/(b-a) & a < x \le b, \\ (c-x)/(c-b) & b < x \le c, \\ 0 & x > c \end{cases} \tag{2.1}$$

$$\Pi(x;a,b,c,d) = \begin{cases} 0 & x \le a, \\ (x-a)/(b-a) & a < x \le b, \\ 1 & b < x \le c, \\ (d-x)/(d-c) & c < x \le d, \\ 0 & x > d \end{cases} \tag{2.2}$$

and

$$\pi(x;b,c) = \begin{cases} \mathrm{S}(x;c-b,c-b/2,c) & x \le c, \\ 1-\mathrm{S}(x;c,c+b/2,c+b) & x > c \end{cases} \tag{2.3}$$

where Zadeh's S-function is defined as

$$\mathrm{S}(x;a,b,c) = \begin{cases} 0 & x \le a, \\ 2(\frac{x-a}{c-a})^2 & a < x \le b, \\ 1-2(\frac{x-c}{c-a})^2 & b < x \le c, \\ 1 & x > c. \end{cases} \tag{2.4}$$

In addition to choosing a membership function shape, we have to construct a set of membership functions for all the fuzzy subsets (linguistic terms) for each input and each output. Let us first define some properties of membership functions: peak point, the symmetric or asymmetric membership function, left and right width, crosspoints, and the crosspoint level, which will be referred to in the later discussions.

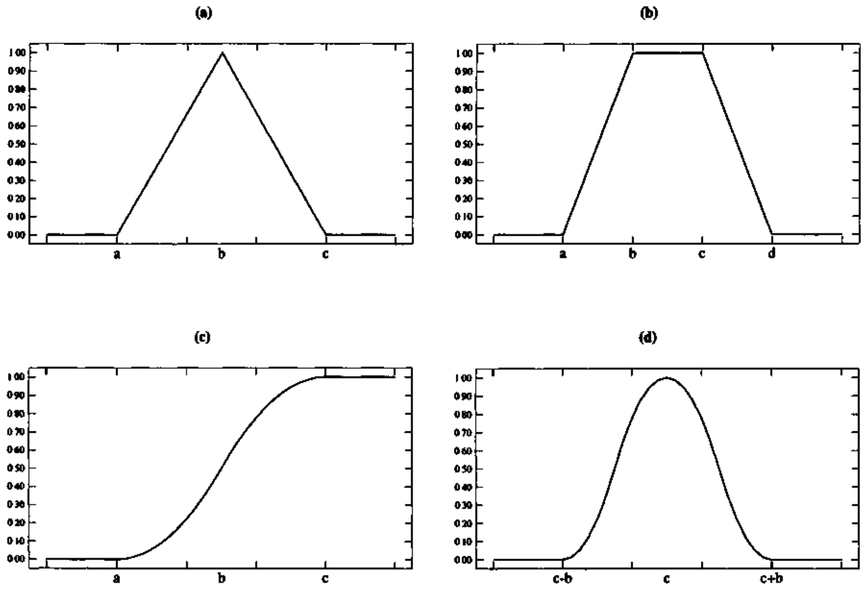


Figure 2.1 (a) A triangular membership function; (b) a trapezoidal membership function; (c) an S-shape membership function; and (d) a bell-shaped membership function.

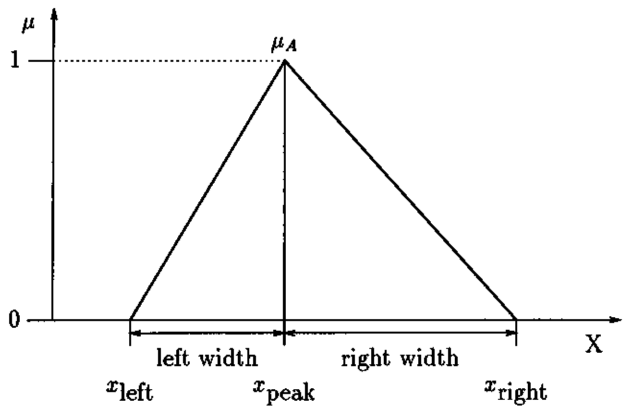


Figure 2.2 The peak point, the left and right width of an asymmetric triangular membership function.

Peak Point

Assuming that μ_A: $\mathbf{X} \to [0, 1]$, the peak point is defined as the value x_{peak} from the domain $\mathbf{X}$ which makes $\mu_A(x_{\text{peak}}) = 1$. Figure 2.2 illustrates the peak point of a triangular membership function. The peak value is an interval for a trapezoidal-shaped membership function.

Left and Right Width

The left width (w_l) is the length of the interval from the peak point to the nearest point in the left x_{left} which has $\mu_A(x_{\text{left}}) = 0$, that is,

$$w_l = x_{\text{peak}} - x_{\text{left}}. \tag{2.5}$$

Similarly, the right width (w_r) is the length of the interval from the peak point to the nearest point in the right x_{right} which has $\mu_A(x_{\text{right}}) = 0$, that is,

$$w_r = x_{\text{right}} - x_{\text{peak}}. \tag{2.6}$$

If $w_l = w_r$ then the membership function is symmetric. Otherwise it is asymmetric. Figure 2.2 illustrates the above notions for an asymmetric triangular membership function.

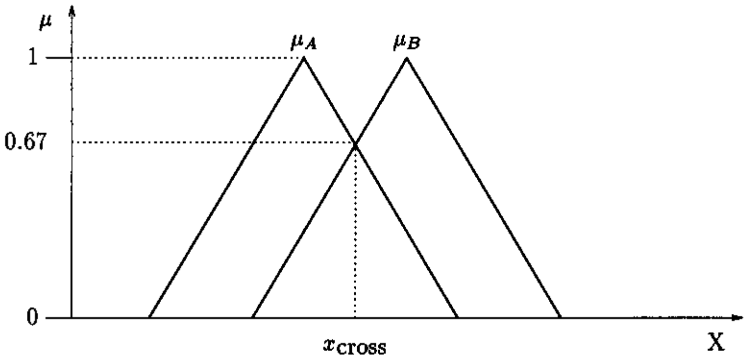


Figure 2.3 The crosspoint and the crosspoint level of triangular membership functions.

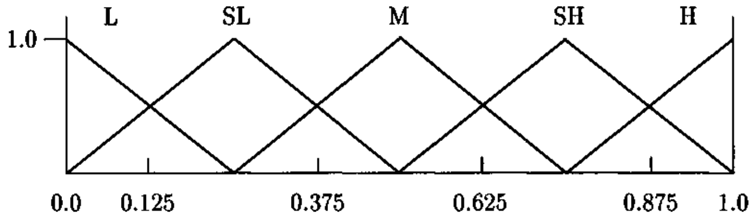
Crosspoints and Crosspoint Level

Figure 2.3 shows two membership functions representing two different linguistic terms, A and B of $\mathbf{X}$. A crosspoint is the value x_{cross} in $\mathbf{X}$ where $\mu_A(x_{\text{cross}}) = \mu_B(x_{\text{cross}}) > 0$ A crosspoint level can then be defined as the membership grade of x_{cross} in either fuzzy subset A or fuzzy subset B (they have the same value by the definition of the crosspoint). Note that two neighboring membership functions may have more than one crosspoint. For the example shown in Fig. 2.3, the crosspoint level is 0.67.

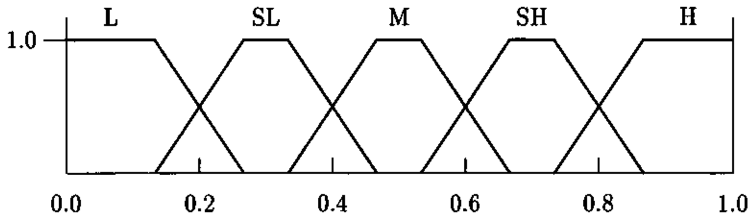
In a fuzzy control application, a set of membership functions are normally chosen based on the following principles:

- The same shape of symmetric membership function is usually applied to all fuzzy subsets of an input or an output.
- These membership functions are evenly distributed in the value range of each input or each output. They usually have a crosspoint level of 0.5.
- Different inputs and outputs can have the same or different shapes of membership functions, and the same or different numbers of linguistic terms.

The value for each variable is usually normalized to the range [-1, 1] or [0, 1]. Figure 2.4 shows the evenly-distributed triangular (a) and trapezoidal (b) membership functions with five linguistic labels, L (low), SL (somehow low), M (medium), SH (somehow high), and H (high).



(a)



(b)

Figure 2.4 Evenly distributed triangular (a) and trapezoidal (b) membership functions with five linguistic labels.

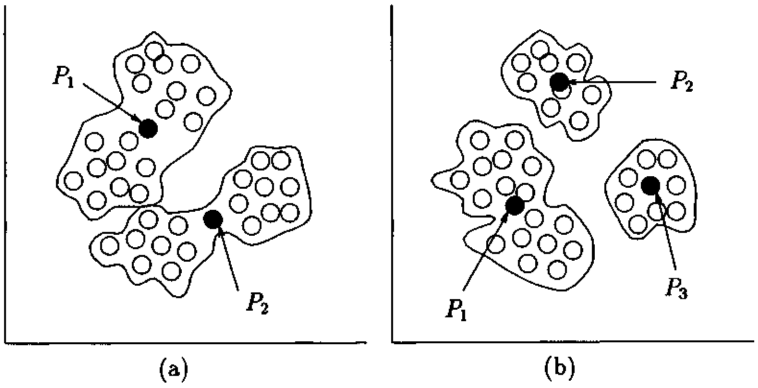


Figure 2.5 The resulting clusters of applying two different similarity measures, (a) the angle difference, and (b) Euclidean distance (P_1, P_2, and P_3 are prototypes).

2.3 Clustering Approaches

Heuristically chosen membership functions do not reflect the actual data distribution in the input and the output spaces. They may not be suitable for fuzzy pattern recognition. To build membership functions from the data available, we can use a clustering technique to partition the data, and then produce membership functions from the resulting clusters.

"*Clustering*" is a process to obtain a partition P of a set E of N objects X_i ($i = 1, 2, ..., N$), by the use of a resemblance or disemblance measure, such as a distance measure d. A *partition* P is a set of disjoint subsets of E and an element P_s of P is called a *cluster* and the centers of the clusters are called centroids or prototypes.

Based on different similarity measures, different clusters are obtained. Figure 2.5 illustrates the clustering results of 2-D inputs when (a) the angle difference defined as

$$\cos(\mathbf{X}_i, \mathbf{X}_j) = \frac{\mathbf{X}_i^T \mathbf{X}_j}{\|\mathbf{X}_i\| \, \|\mathbf{X}_j\|}, \tag{2.7}$$

and (b) Euclidean distance defined as

$$d(\mathbf{X}_i, \mathbf{X}_j) = \sqrt{\textstyle\sum_k (x_{ik} - x_{jk})^2} \tag{2.8}$$

are used for the similarity measure, respectively.

Many techniques have been developed for clustering data. In the following sections, we will introduce three of these techniques, c-means clustering, adaptive vector quantization (AVQ), and the self-organizing map (SOM).

2.3.1 C-means Clustering Approach

C-means clustering is a simple unsupervised learning method which can be used for data grouping or classification when the number of clusters is known. It consists of the following steps:

1. Choose the number of clusters, k.

2. Set initial centers of clusters, $\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_k$ to the arbitrarily selected k vectors from the training set.

3. Classify each vector $\mathbf{x}_l = [x_{l1}, x_{l2}, ..., x_{ln}]^T$ (n is the dimension of input vectors) into the closest center $\mathbf{c}_i$ by Euclidean distance measure:

$$\| \mathbf{x}_l - \mathbf{c}_i \| = \min_j \| \mathbf{x}_l - \mathbf{c}_j \| . \tag{2.9}$$

4. Recompute the estimates for the cluster centers $\mathbf{c}_i$. Let $\mathbf{c}_i = [c_{i1}, c_{i2}, ..., c_{in}]^T$, c_{im} is computed by

$$c_{im} = \frac{\sum_{\mathbf{x}_{l_i} \in \text{ cluster } i} x_{l,m}}{N_i} \tag{2.10}$$

   where N_i is the number of vectors in the ith cluster.

5. If none of the cluster centers ($\mathbf{c}_i$, $i = 1, ..., k$) changes in Step 4, stop; otherwise go to step 3.

Note that we do not make use of class labels so the c-means clustering is an unsupervised learning algorithm. Variances ($\mathbf{v}_i, i = 1, 2, ..., k$) can be computed for all clusters resulting from the c-means clustering. Let $\mathbf{v}_i = [v_{i1}, v_{i2}, ..., v_{in}]^T$, then

$$v_{im} = \sqrt{\frac{\sum_{\mathbf{x}_{l_i} \in \text{ cluster } i} (x_{l,m} - c_{im})^2}{N_i}}. \tag{2.11}$$

We can make use of the c-means cluster centers and variances, which reflect the actual data distribution in the input space, to generate membership functions for each input. The idea is to approximate each cluster as a hyper-ellipsoid with its center being the cluster center and the lengths of axes decided by the corresponding variance of the cluster. The projection of a hyper-ellipsoid onto each axis will producing a symmetric triangular membership function with the peak point being the corresponding component of the cluster center. Figure 2.6 demonstrates membership function generation for a problem of two inputs and seven clusters. As shown in the figure, Cluster 1 was approximated by an ellipsoid with lengths of two axes being r_{11} and r_{12}, respectively. For a general problem with n inputs, we have n axes of lengths: $r_{i1}, r_{i2}, ..., r_{in}$ for cluster i. We define r_{im} as

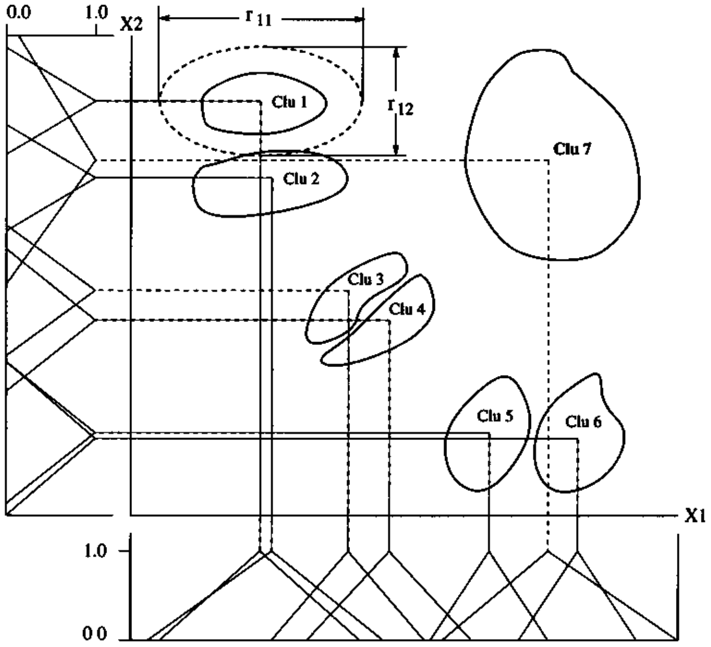


Figure 2.6 Deriving membership functions from the c-means cluster centers and variances.

$$r_{im} = 2av_{im} \tag{2.12}$$

where $1 \leq a \leq 4$ is a factor which decides the degree of overlapping among neighboring clusters which actually reflects the fuzziness of the data. Each ellipsoid is projected onto each input axis ($m = 1, 2, ..., n$) generating a triangular membership function with the peak in the center of each cluster, $\Lambda(x_m; c_{im} - r_{im}/2, c_{im}, c_{im} + r_{im}/2)$. Note that this triangular membership function is actually a special case of trapezoidal functions: $\Pi(x_m; c_{im} - r_{im}/2, c_{im}, c_{im}, c_{im} + r_{im}/2)$. We can also see from Fig. 2.6 that some neighboring clusters are so close to each other so that their projections on one axis (could be on two axes) have a high degree of overlapping (a high crosspoint level in the corresponding membership functions), for example, clusters 1 and 2 on axis $X1$, and clusters 5 and 6 on axis $X2$. In Section 2.3.4, we shall discuss a merging scheme to combine those neighboring membership functions with a very high crosspoint level. Membership functions learned from the c-means clustering algorithm have been applied to geographical map image segmentation which will be reported in Section 6.7.1.

2.3.2 Learning with Adaptive Vector Quantization

The c-means clustering algorithm produces the grouping of input patterns and corresponding cluster centers. However, instead of learning, the variances of clusters have to be computed after the clustering process. In this section, we will discuss a clustering method which can learn cluster covariance matrixes as well as cluster centers.

Dickerson and Kosko proposed a technique to learn fuzzy functions by using adaptive vector quantization (AVQ) [23]. In their approach, the unsupervised AVQ competitive learning was used to estimate the local centroids and covariance matrixes of clusters in the input-output space. The covariance matrix of each quantization vector defines an ellipsoid with its center being the cluster centroid. Fuzzy patches or rules were learned from the resulting ellipsoids.

In our application, we used the AVQ algorithm to learn the membership functions of input variables. The AVQ is a neural system shown in Fig. 2.7 which uses unsupervised competitive learning. The AVQ system learns the synaptic connection matrix $\mathbf{M} = [\mathbf{m}_1, \mathbf{m}_2, ..., \mathbf{m}_p]$ (p is the number of output neurons) between the input neurons in layer $\mathbf{F_X}$ and the output neurons in layer $\mathbf{F_Y}$ which will be the first-order estimation of the unknown probability density function of the data. The following stochastic difference equation is used to estimate $\mathbf{m}_i = [m_{i1}, m_{i2}, ..., m_{in}]^T$ (n is the number of inputs):

$$\mathbf{m}_i(k+1) = \begin{cases} \mathbf{m}_i(k) + \eta_k[\mathbf{x}_k - \mathbf{m}_i(k)] & \text{if the } i\text{th neuron wins,} \\ \mathbf{m}_i(k) & \text{if the } i\text{th neuron loses} \end{cases} \tag{2.13}$$

where η_k is a learning coefficient given by

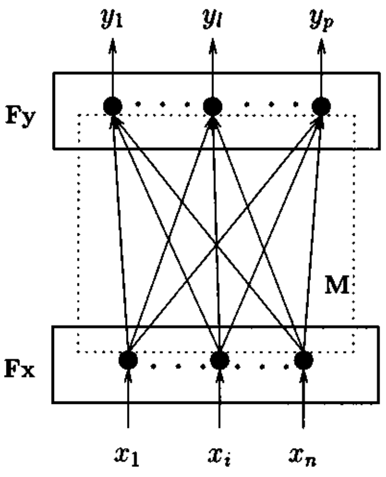


Figure 2.7 The AVQ architecture.

$$\eta_k = 0.1(1 - \frac{k}{k_{max}}). \tag{2.14}$$

The synaptic vectors converge to the pattern centroids. On the other hand, the local covariance matrix ($\mathbf{V}$) that give a second-order estimate are learned by

$$\mathbf{V}_i(k+1) = \begin{cases} \mathbf{V}_i(k) + \beta_k[(\mathbf{x}_k - \mathbf{m}_i(k))(\mathbf{x}_k - \mathbf{m}_i(k))^T - \mathbf{V}_i(k)] & \\ & \text{if the } i\text{th neuron wins,} \\ \mathbf{V}_i(k) & \text{if the } i\text{th neuron loses} \end{cases} \tag{2.15}$$

where β_k is again a learning coefficient which is similar to η_k. Both learning coefficients are decreasing functions of time.

The competitive AVQ algorithm consists of the following steps [23]:

1. Set initial $\mathbf{m}_i(0)$ $(i = 1, ..., p)$ to arbitrary p input vectors ($\mathbf{m}_i(0)$ can be set to the cluster centers obtained from the c-means clustering in order to speed up the convergence).

2. For a sample $\mathbf{x}_k$, find the closest or "winning" synaptic vector $\mathbf{m}_i(k)$ based on the Euclidean distance measure:

$$\| \mathbf{x}_k - \mathbf{m}_i(k) \| = \min_j \| \mathbf{x}_k - \mathbf{m}_j(k) \| . \tag{2.16}$$

3. Update the winning synaptic vector $\mathbf{m}_i(k)$ according to Eq. (2.13) and its local covariance matrix estimate $\mathbf{V}_i(k)$ according to Eq. (2.15).

The data are statistically normalized so that all inputs have equal weight in the distance measure.

The covariance ellipsoid is the locus of all $\mathbf{x}s$ that satisfy

$$c^2 = (\mathbf{x} - \mathbf{m}_i)^T \mathbf{V}_i^{-1} (\mathbf{x} - \mathbf{m}_i). \tag{2.17}$$

The eigenvalues of $\mathbf{V}_i^{-1}$ are $\lambda_{i1}, ..., \lambda_{in}$. The unit eigenvectors define the *direction cosines* for each axis of the ellipse. If the jth eigenvector is $\mathbf{e}_{ij} = [e_{ij1}, e_{ij2}, ..., e_{ijn}]^T$, then the direction cosine of eigenvector j to axis l is $\cos\gamma_{ijl}$ $(1 \leq j, l \leq n)$ which is given by

$$\cos\gamma_{ijl} = e_{ijl}/||\mathbf{e}_{ij}||. \tag{2.18}$$

The Euclidean half-lengths of the axes are equal to $c/\sqrt{\lambda_{i1}}$, $c/\sqrt{\lambda_{i2}}$, ..., $c/\sqrt{\lambda_{in}}$. Therefore, the projection of the ith hyper-ellipsoid onto the lth axis is the interval δ_{il} which is given by (see Fig. 2.8 for a 2-D case)

$$\begin{aligned} \delta_{il} &= [m_{il} - c(|\cos\gamma_{i1l}|/\sqrt{\lambda_{i1}} + ... + |\cos\gamma_{inl}|/\sqrt{\lambda_{in}}), \\ &\quad m_{il} + c(|\cos\gamma_{i1l}|/\sqrt{\lambda_{i1}} + ... + |\cos\gamma_{inl}|/\sqrt{\lambda_{in}})] \end{aligned} \tag{2.19}$$

where $1 \leq c \leq 4$. The generated membership functions are symmetric and peaked at the pattern centroids. As it occured in the membership functions generated from the c-means clusters, some neighboring membership functions with a high degree of overlapping can be merged with little effect on the system performance. The membership functions learned by the AVQ algorithm have also been applied to geographical map image segmentation [25].

2.3.3 Self-organizing Map Approach

Kohonen self-organizing map (SOM) uses unsupervised learning algorithm to modify the weight vectors ($\mathbf{W}_i$, $i = 1, 2, ..., m$, and m is the total number of nodes in the map) of a network to model the features found in the training data [30] (see Fig. 2.9). A topographic map is automatically organized by a cyclic process of comparing input patterns to weight vectors for each node. The weight vector to which inputs match is selectively optimized to represent an average of the training data. As a result, all the training data are represented by the weight vectors of the map which can be considered as the prototypes or centroids of the input patterns.

Staring with a randomly organized set of nodes, and proceeding to the creation of a feature map representing the prototypes of the input data, The training procedure is as follows:

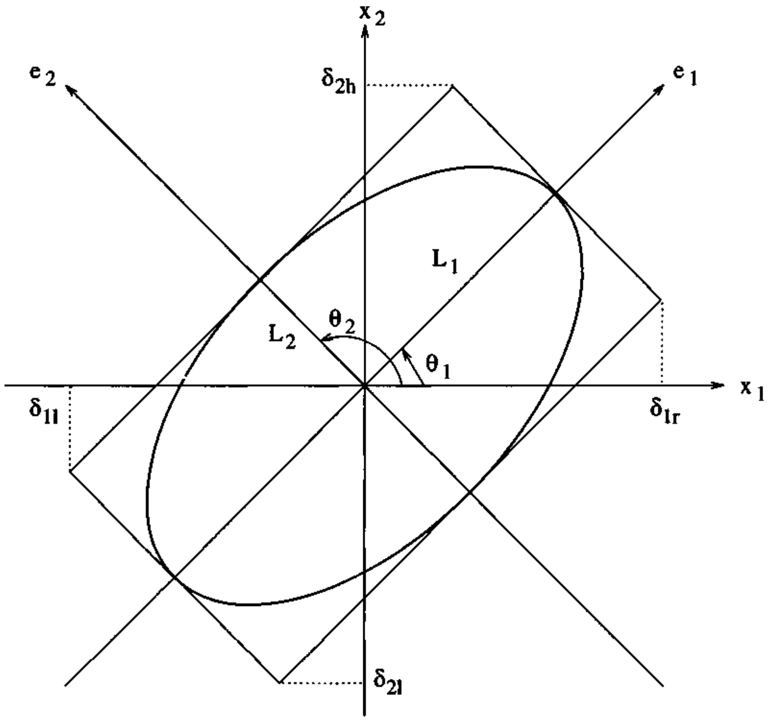


Figure 2.8 Schematic demonstration of projection of an ellipsoid on axes X_1 and X_2 (L_1 and L_2 are Euclidean half-lengths with $L_1 = c/\sqrt{\lambda_1}$ and $L_2 = c/\sqrt{\lambda_2}$; $\delta_{1r} = \delta_{1l} = L_1|\cos\theta_1| + L_2|\cos\theta_2|$ and $\delta_{2h} = \delta_{2l} = L_1|\cos(\pi/2 - \theta_1)| + L_2|\cos(\pi/2 - \theta_2)|$).

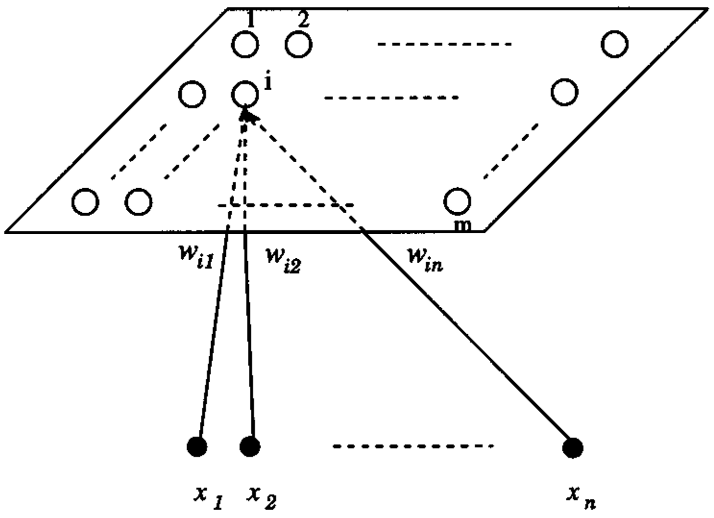


Figure 2.9 One-layer two-dimensional Kohonen self-organizing map structure (m: the total number of nodes in the map; $(x_1, x_2, ..., x_n)$: the inputs; and $\mathbf{W}_i = [w_{i1}, w_{i2}, ..., w_{in}]^T$: the weight vector for node i).

1. Initialize the weights w_{ij} ($1 \leq i \leq m, 1 \leq j \leq n$) to small random values, where n is the number of inputs. Set the initial radius of the neighborhood around node i as $N_i(0)$.

2. Present the inputs $x_1(t), x_2(t), ..., x_n(t)$, where $x_j(t)$ is the jth input at time t.

3. Calculate the distance d_i between the inputs and node i by

$$d_i = \sum_{j=1}^{n}(x_j(t) - w_{ij}(t))^2. \tag{2.20}$$

4. Determine i_* which minimizes d_i.

5. Update the weights for node i_* and its neighbors in $N_{i_*}(t)$. The new weights for i in $N_{i_*}(t)$ are

$$w_{ij}(t+1) = w_{ij}(t) + a(t)(x_j(t) - w_{ij}(t)) \tag{2.21}$$

   where $a(t)$ is the learning rate. Both $a(t)$ and $N_{i_*}(t)$ decrease as t increases.

6. If the process reaches the maximum number of iterations, stop; otherwise, go to Step 2.

The self-organizing algorithm has a characteristics of preserving the topology in the sense that the nodes located physically next to each other will respond to classes of input vectors that are likewise next to each other. It is an advantage to use a 2-D map in which we can easily visualize the relationship among nodes. However, the 1-D or higher dimensional maps are also used for certain applications. Shown in Fig. 2.10 is a 15 × 15 self-organizing map obtained from a set of 10,426 handwritten digit samples. As we can see from the map, those handwritten digits which are similar in appearance are close to one another. For example, some of prototypes of "7" are very similar to those of "9" or "1" in appearance and this is truly indicated in the map. The same is observed for some prototypes of "2" and "3". This topology-preserving property of the self-organizing map is in much demanded for generating more robust membership functions, in particular, when the membership function merging will be performed. We shall discussion an approach for the membership function merging in Section 2.3.4.

We use the SOM prototypes and variances to determine the membership functions for each input. Each prototype generates a triangular membership function for each input variable with the peak of the function located at each component of the prototype. Widths of fuzzy subsets are determined by the corresponding variances. Suppose that m prototypes are $\mathbf{W}_i$ ($i = 1, 2, ..., m$) with the corresponding variances $\mathbf{V}_i$ ($i = 1, 2, ..., m$). Assume that $\mathbf{W}_i = [w_{i1}, w_{i2}, ..., w_{in}]^T$ and $\mathbf{V}_i = [v_{i1}, v_{i2}, ..., v_{in}]^T$. The centers of the triangular membership functions on the jth axis are $w_{1j}, w_{2j}, ..., w_{mj}$. The corresponding regions are set to

4	9	4	4	7	7	7	9	9	9	8	8	8	4	4
4	4	4	4	9	7	7	9	9	9	8	8	1	1	4
4	4	4	4	9	7	7	7	9	7	9	1	1	1	7
4	4	4	4	4	2	7	7	9	7	1	1	1	4	2
5	5	4	4	4	9	7	7	7	1	1	1	8	1	5
5	5	8	9	8	8	7	7	7	1	1	1	9	8	0
5	5	5	8	8	8	8	7	1	1	1	6	5	6	0
0	5	8	8	8	8	8	2	2	8	6	6	6	0	0
0	0	0	8	8	8	2	2	2	6	6	6	0	0	0
0	0	5	5	8	8	2	2	2	2	6	6	0	0	0
0	0	0	5	5	3	3	2	2	2	6	6	6	0	0
0	0	0	5	3	3	3	3	2	2	2	6	6	6	6
6	0	5	3	3	3	3	3	2	2	2	2	6	6	6
6	8	9	3	3	3	3	3	3	2	2	2	2	6	4
4	9	9	7	7	3	3	3	8	2	2	2	2	2	4

Figure 2.10 A 15×15 self-organizing map obtained from a set of 10,426 handwritten digit samples.

$$\begin{aligned}
&[w_{1j} - av_{1j}, w_{1j} + av_{1j}], \\
&[w_{2j} - av_{2j}, w_{2j} + av_{2j}], \\
&\ldots\ldots, \\
&[w_{mj} - av_{mj}, w_{mj} + av_{mj}].
\end{aligned}$$

Parameter a ($1 \leq a \leq 4$) is a scaling factor which reflects the fuzziness of the data. A larger a is used for more uncertain data. The membership functions generated from the SOM maps have found applications in handwritten digit recognition [26], [27].

2.3.4 Merging Neighboring Membership Functions

Suppose that input k has l_k linguistic labels. The maximum number of fuzzy rules that can be learned is $\prod_{k=1}^{n} l_k$. In pattern recognition application, we can easily get a large number of training examples so that the total of $(\prod_{k=1}^{n} l_k)$ fuzzy rules could possibly be produced. By reducing the number of linguistic labels for each input, we can reduce the number of produced fuzzy rules. One way to do this is to combine two neighboring membership functions with a great degree of overlapping (a very high crosspoint level). If two neighboring membership functions (considered as trapezoidal membership functions for a general case) are $\Pi(x_j; a_{i-1}, b_{i-1}, c_{i-1}, d_{i-1})$ and $\Pi(x_j; a_i, b_i, c_i, d_i)$, they will be merged if the following condition is satisfied:

$$\left| \frac{b_i + c_i}{2} - \frac{b_{i-1} + c_{i-1}}{2} \right| \leq l_T \tag{2.22}$$

where l_T is a pre-specified threshold. The new membership function after the combination is $\Pi(x_j; a_{i*}, b_{i*}, c_{i*}, d_{i*})$, with the following parameters:

$$\begin{aligned}
a_{i*} &= \min(a_i, a_{i-1}), \\
b_{i*} &= \min(b_i, b_{i-1}), \\
c_{i*} &= \max(c_i, c_{i-1}), \\
d_{i*} &= \max(d_i, d_{i-1}).
\end{aligned} \tag{2.23}$$

As a result of the merging process, some membership functions have trapezoidal shapes instead of triangular ones (see Fig. 2.11).

Figure 2.6 shows that Clusters 3 and 4 are so close that they can be combined into one cluster. Accordingly, membership functions on two axes derived from these two clusters are also merged. However, for the case of Clusters 1 and 2, and the case of Clusters 5 and 6, they are merged only on one axis (the horizontal axis and the vertical axis respectively). In most merging cases for high-dimensional input vectors, the membership functions of at least one input are separated and, therefore, the discrimination ability for two clusters is not affected by this merging. Although using a smaller number of clusters can also reduce the number of fuzzy rules, it loses the discrimination ability for the two sub-clusters.

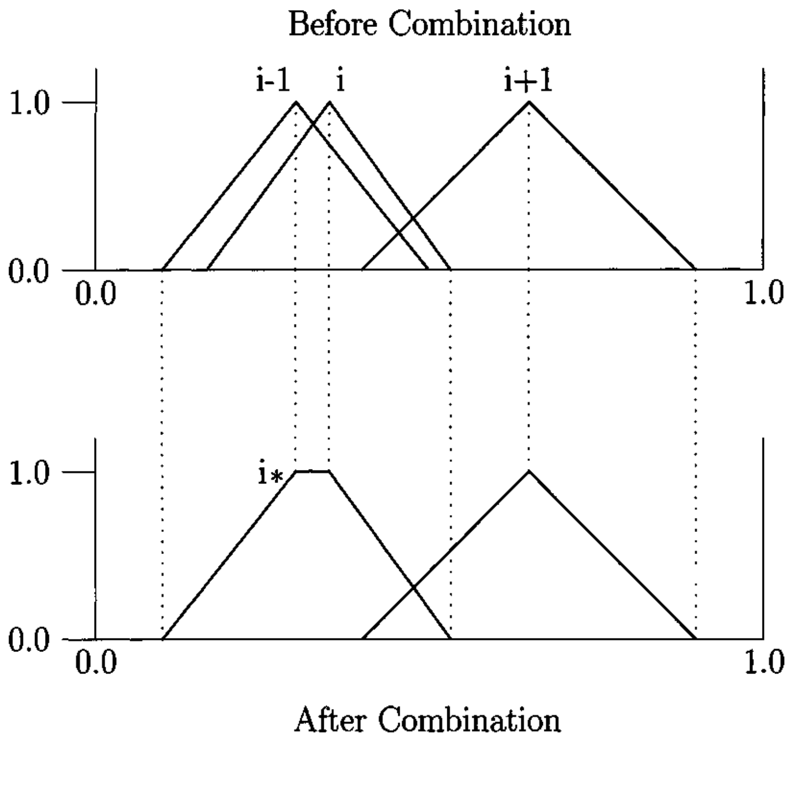


Figure 2.11 Schematic demonstration of merging neighboring membership functions.

The membership functions generated by the clustering technique and the merging scheme discussed above have the following characteristics:

- A set of mixed triangular and trapezoidal membership functions are generated for each input.
- The membership functions can be symmetric or asymmetric.
- Different inputs usually have different numbers of linguistic terms.
- The crosspoint levels (x_{cross}) can be any values in the range of [0, 1].

2.4 Tuning of Membership Functions

As discussed in the previous sections, membership functions can be chosen heuristically or generated by using one of the clustering techniques. The heuristically chosen or generated membership functions can be tuned to improve their performance by using a set of training data. The tuning of membership functions becomes more important if the merging procedure has been taken to reduce the number of fuzzy subsets or a minimization technique has been used to reduce the number of fuzzy rules. Two techniques, known as the gradient descent algorithm and a neural network approach for tuning membership functions are discussed in the following sections.

2.4.1 Gradient Descent Algorithm

Driankov *et al* presented a membership function tuning approach based on the gradient descent algorithm [28]. In this section, we present a modified version of the approach for classification problems. Assume that a set of symmetric triangular membership functions are used. As discussed in Section 2.2, a triangular membership function $\Lambda(x; a, b, c)$ (symmetric or asymmetric) can be determined by three parameters a, b, and c. However, only two parameters, the peak point a_{ji} and the width b_{ji} have to be used to represent a symmetric triangular membership function as shown in Fig. 2.12. The symmetric triangular membership function is thus given by

$$m_{ji}(x_j) = \begin{cases} 1 - 2\frac{|x_j - a_{ji}|}{b_{ji}} & \text{if } a_{ji} - b_{ji}/2 \leq x_j \leq a_{ji} + b_{ji}/2, \\ 0 & \text{otherwise.} \end{cases} \tag{2.24}$$

Defuzzification is a process of producing non-fuzzy output(s) from a subset of fired fuzzy rules . A non-fuzzy output is a function of the firing degrees and the fuzzy outputs of fired fuzzy rules regardless of what defuzzification method is used. The firing degree of a fuzzy rule is itself a function of membership function parameters and inputs. For triangular membership functions, the non-fuzzy output O is partly decided by a_{ij}s and b_{ij}s. When the centroid defuzzification method is adopted, O is given by

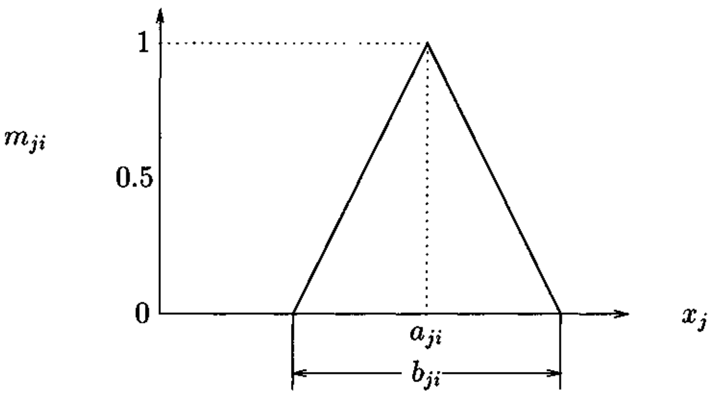


Figure 2.12 A symmetric triangular membership function.

$$O = \frac{\sum_{i=1}^{K} D^i O^i}{\sum_{i=1}^{K} D^i} \tag{2.25}$$

where K is the number of rules, O^i is the fuzzy output of rule i. For the L-class classification problem, O^i takes values of 0, 1, ..., $(L-1)$. D^i measures how an input vector matches the antecedent conditions (IF-part) of the ith rule.

$$D^i = \prod_{l=1}^{n} m_{li}(x_l) \tag{2.26}$$

where n is the number of inputs and m_{li} is the membership grade of input l in the fuzzy subset that the ith rule takes. Obviously,

$$m_{li}(x_l) = f(a_{li}, b_{li}, x_l). \tag{2.27}$$

If a set of training data that describes the desirable output, O^d, for inputs, x_1, x_2,..., x_n, is available, then the fuzzy rule-based system can be optimized by minimizing the error between the output given by the defuzzification based on Eq. (2.25) and the desired output given by the training data. We can define an error function, E, as

$$E = \frac{1}{2}(O - O^d)^2 \tag{2.28}$$

where (1/2) is introduced for the simplicity in the following discussion.

The steepest descent algorithm, which is the simplest method for solving the optimization problem, can be adopted to minimize the error function. The steepest descent algorithm states that from any point the objective function (the error function in this case) decreases most rapidly in the direction of the negative gradient vector of its parameters at that point. For the error function $E(Z)$ with a parameter set of $Z = (z_1, z_2, ..., z_p)$, the negative gradient vector is

$$\left(-\frac{\partial E}{\partial z_1}, -\frac{\partial E}{\partial z_2}, ..., -\frac{\partial E}{\partial z_p}\right).$$

If $z_i(t)$ is the value of the ith parameter at iteration t, the modification of z_i is given by

$$z_i(t+1) = z_i(t) - \eta \frac{\partial E(Z)}{\partial z_i}, \quad i = 1, 2, ..., p \tag{2.29}$$

where η is the constant (learning rate) which decides how much the parameters are modified at each iteration. The error function will eventually converge to a minimum value after a number of iterations.

For membership function tuning, the variables to be modified in the error function are the membership function parameters a_{ji} and b_{ji}, i. e.,

$$Z = (a_{11}, ..., a_{nK}, b_{11}, ..., b_{nK}). \tag{2.30}$$

The maximum number of parameters to be optimized is $(n \times K)$. It could be a large number if many fuzzy rules are available for a many-input problem. In practice, it is quite often that different rules share certain fuzzy subsets (linguistic terms) for certain inputs, so the actual number of parameters to be modified is smaller.

Substituting Eqs. (2.26) and 2.25 into Eq. (2.28), we have

$$E = \frac{1}{2}\left(\frac{\sum_{k=1}^{K}(\prod_{l=1}^{n} m_{lk}(x_l))O^k}{\sum_{k=1}^{K}(\prod_{l=1}^{n} m_{lk}(x_l))} - O^d\right)^2. \tag{2.31}$$

The steepest descent algorithm gives the following iteration equations for the parameters a_{ji} and b_{ji}:

$$a_{ji}(t+1) = a_{ji}(t) - \eta_a \frac{\partial E}{\partial a_{ji}}, \quad j = 1, ..., n;\ i = 1, ..., K, \tag{2.32}$$

$$b_{ji}(t+1) = b_{ji}(t) - \eta_b \frac{\partial E}{\partial b_{ji}}, \quad j = 1, ..., n;\ i = 1, ..., K. \tag{2.33}$$

Calculating the partial derivative of $(\frac{\partial E}{\partial a_{ji}})$ yields the following equation,

$$a_{ji}(t+1) = a_{ji}(t) - \eta_a \frac{2D^i}{b_{ji} m_{ji}(x_j) \sum_{k=1}^{K} D^k}[O(t) - O^d][O^i - O(t)]\mathrm{sgn}[x_j - a_{ji}(t)]. \tag{2.34}$$

where "sgn" is the sign operator which is defined as

$$\mathrm{sgn}(x) = \begin{cases} -1 & \text{if } x \leq 0, \\ 1 & \text{otherwise.} \end{cases} \tag{2.35}$$

Let $\eta_a' = 2\eta_a$ and $c_{ji}(t) = \frac{D^i}{b_{ji}(t) m_{ji}(x_j) \sum_{k=1}^{K} D^k}$, then we have

$$a_{ji}(t+1) = a_{ji}(t) - \eta_a' c_{ji}(t)[O(t) - O^d][O^i - O(t)]\mathrm{sgn}[x_j - a_{ji}(t)]. \tag{2.36}$$

Similarly, we have

$$b_{ji}(t+1) = b_{ji}(t) - \eta_b c_{ji}(t)[O(t) - O^d][O^i - O(t)][1 - m_{ji}(x_j)]. \tag{2.37}$$

The Tuning Procedure

If a set of training data ($\mathbf{x}_r = [x_{r1}, ..., x_{rn}]^T$, O_r^d, $r = 1, 2, ..., N$) have been collected, we can use the optimization procedure described below to tune triangular membership functions:

1. Set $t = 1$, $p = 1$, $E_T^{\text{old}} = 0$, and $E_T^{\text{new}} = 0$.
2. Calculate the matching degree of an input-output pair for each rule, D^i using Eq. (2.26), and the actual output O using Eq. (2.25).
3. Update parameters a_{ji} and b_{ji} using Eqs. (2.36) and (2.37).
4. Calculate the error $E = \frac{1}{2}(O - O^d)^2$ in order to obtain $E_T^{\text{old}} \leftarrow E_T^{\text{old}} + E$.
5. If $p < N$, set $p \leftarrow p + 1$ and go to Step 2. Otherwise, go to Step 6,
6. If $|E_T^{\text{new}} - E_T^{\text{old}}|$ is smaller than a pre-specified small value, stop. Otherwise, set $p = 1$, $E_T^{\text{new}} = E_T^{\text{old}}$, and $E_T^{\text{old}} = 0$, and go to Step 2.

Note that different types of membership functions and different defuzzification methods require different modification formulae. However, the principle is the same.

2.4.2 Neural Network Approach

The tuning of membership functions is an optimization process. Recently neural networks have been applied for various optimization problems. Figure 2.13 shows a four-layer feedforward neural network which can be used to tune the membership functions and to optimize the defuzzification parameters at the same time [29].

Suppose that we have two inputs x_1 and x_2. The neural network will classify the input patterns into two classes C_1 and C_2. In the neural network shown in

Fig. 2.13, Layer 1 is the input layer. Each node in Layer 2 is associated with a parameterized membership function. In most occasions, a bell-shaped membership function as described in [29] is used.

$$\mu_A(x_i) = \frac{1}{1 + [(\frac{x_i - c_i}{a_i})^2]^{b_i}} \tag{2.38}$$

where x_i is the ith input variable, A is the linguistic label associated with this node function, and $\{a_i,\ b_i,\ c_i\}$ are a set of parameters for the bell-shaped membership function (see Fig. 2.14). Note that $2a_i$ is always the width between two half-height (0.5 membership grade) points no matter what b_i and c_i are.

Assume that there are three linguistic labels, S (Small), M (Medium), and L (Large), for each input as shown in Fig. 2.13. Layer 2 consists of six nodes with each associated with a bell-shaped membership function. Each node in Layer 3 is associated with a rule. There are nine (3^2) rules available, so there are nine nodes in Layer 3. Each node in Layer 3 generates a signal corresponding to the matching degree of a rule. Nine rules are listed below:

R_1: IF x_1 is S AND x_2 is S, THEN the pattern belongs to class $\omega(R_1)$.
R_2: IF x_1 is S AND x_2 is M, THEN the pattern belongs to class $\omega(R_2)$.
R_3: IF x_1 is S AND x_2 is L, THEN the pattern belongs to class $\omega(R_3)$.
R_4: IF x_1 is M AND x_2 is S, THEN the pattern belongs to class $\omega(R_4)$.
R_5: IF x_1 is M AND x_2 is M, THEN the pattern belongs to class $\omega(R_5)$.
R_6: IF x_1 is M AND x_2 is L, THEN the pattern belongs to class $\omega(R_6)$.
R_7: IF x_1 is L AND x_2 is S, THEN the pattern belongs to class $\omega(R_7)$.
R_8: IF x_1 is L AND x_2 is M, THEN the pattern belongs to class $\omega(R_8)$.
R_9: IF x_1 is L AND x_2 is L, THEN the pattern belongs to class $\omega(R_9)$.

where $\omega(R_i) = (C_1 \wedge \neg C_2) \vee (C_2 \wedge \neg C_1)$.

If the multiplication operation is used to compute the matching degree, the output of the ith node in Layer 3, D^i, is given by Eq. (2.26). In this example,

$$m_{11}(x_1) = m_{12}(x_1) = m_{13}(x_1) = \mu_S(x_1),$$

$$m_{14}(x_1) = m_{15}(x_1) = m_{16}(x_1) = \mu_M(x_1),$$

$$m_{17}(x_1) = m_{18}(x_1) = m_{19}(x_1) = \mu_L(x_1),$$

$$m_{21}(x_2) = m_{24}(x_2) = m_{27}(x_2) = \mu_S(x_2),$$

$$m_{22}(x_2) = m_{25}(x_2) = m_{28}(x_2) = \mu_M(x_2),$$

$$m_{23}(x_2) = m_{26}(x_2) = m_{29}(x_2) = \mu_L(x_2).$$

Note that although the same linguistic terms are used for both inputs x_1 and x_2 in this example, the associated membership functions usually take different parameters.

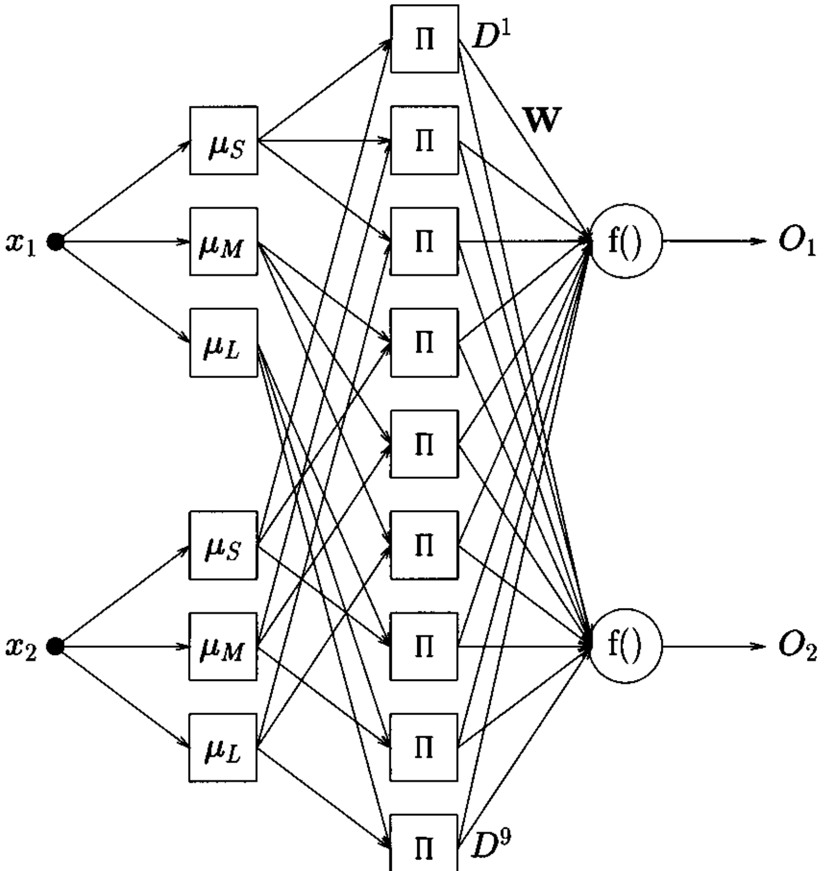


Figure 2.13 A neural network model for membership function tuning.

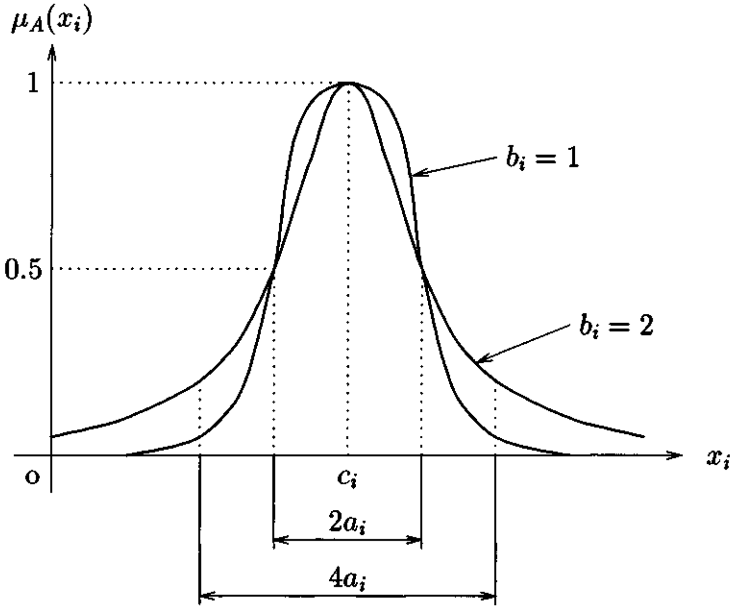


Figure 2.14 Bell-shaped membership functions with different parameters.

We take the linear combination of the matching degrees of the rules and apply a sigmoidal function for each node at Layer 4 to calculate the class membership. The jth output of Layer 4 (also the output of the neural network) is

$$O_j = f(\sum_{i=1}^{9} w_{ji} D^i), \quad j = 1, 2 \tag{2.39}$$

where w_{ji} is a measure of the contribution of the ith rule to class j. They are defuzzification parameters and can be optimized at the same time of the tuning process. The sigmoidal transfer function $f(x)$ is defined as

$$f(x) = \frac{1}{1 + \exp(-x)}. \tag{2.40}$$

Now let us consider a general case. Suppose that there are K rules, n inputs, and L classes. The membership function of the jth input for the ith rule is m_{ji} which is defined as

$$m_{ji}(x_j) = \frac{1}{1 + [(\frac{x_j - c_{ji}}{a_{ji}})^2]^{b_{ji}}}. \tag{2.41}$$

The output of the ith node of Layer 3, D^i, is given by

$$D^i = \prod_{j=1}^{n} m_{ji}(x_j). \tag{2.42}$$

Let $z_l = \sum_{i=1}^{K} w_{li} D^i$, and finally, the output of the lth node of Layer 4, O_l, is given by

$$O_l = f(z_l) = f(\sum_{i=1}^{K} w_{li} D^i). \tag{2.43}$$

The error function is again defined as

$$E = \frac{1}{2} \sum_{l=1}^{L} (O_l - O_l^d)^2 \tag{2.44}$$

where O_l^d is the desired output for the lth output node and E is a function of membership function parameters a_{ji}, b_{ji}, and c_{ji} ($j = 1, 2, ..., n$ and $i = 1, 2, ..., K$), and defuzzification parameters w_{li} ($l = 1, 2, .., L$).

Based on the steepest descent algorithm, these parameters can be modified by

$$a_{ji}(t+1) = a_{ji}(t) - \eta_a \frac{\partial E}{\partial a_{ji}}, \tag{2.45}$$

$$b_{ji}(t+1) = b_{ji}(t) - \eta_b \frac{\partial E}{\partial b_{ji}}, \tag{2.46}$$

$$c_{ji}(t+1) = c_{ji}(t) - \eta_c \frac{\partial E}{\partial c_{ji}}, \tag{2.47}$$

$$w_{li}(t+1) = w_{li}(t) - \eta_w \frac{\partial E}{\partial w_{li}} \tag{2.48}$$

where η_a, η_b, η_c, and η_w are the learning rates which control the magnitudes for the modification of these parameters at each iteration.

By calculating these partial derivatives, we have

$$\frac{\partial E}{\partial w_{li}} = -\delta_{ol} D^i \tag{2.49}$$

where $\delta_{ol} = f'(z_l)(O_l^d - O_l)$. For a sigmoidal transfer function,

$$f'(z_l) = f(z_l)[1 - f(z_l)] = O_l(1 - O_l). \tag{2.50}$$

Similarly, we have

$$\frac{\partial E}{\partial a_{ji}} = -\frac{2b_{ji}}{a_{ji}} \delta_{ri} D^i \tag{2.51}$$

where

$$\delta_{ri} = [1 - m_{ji}(x_j)] \sum_{l=1}^{L} \delta_{ol} w_{li}, \tag{2.52}$$

and

$$\begin{aligned}
\frac{\partial E}{\partial b_{ji}} &= \ln[(\frac{x_j - c_{ji}}{a_{ji}})^2] \delta_{ri} D^i, && (2.53) \\
\frac{\partial E}{\partial c_{ji}} &= -\frac{2b_{ji}}{x_j - c_{ji}} \delta_{ri} D^i. && (2.54)
\end{aligned}$$

The parameters are then modified by

$$\begin{aligned}
w_{li}(t+1) &= w_{li}(t) + \eta_w \delta_{ol} D^i, && (2.55) \\
a_{ji}(t+1) &= a_{ji}(t) + \eta_a \frac{2b_{ji}(t)}{a_{ji}(t)} \delta_{ri} D^i. && (2.56)
\end{aligned}$$

Let $\eta_a' = 2\eta_a$, then

$$a_{ji}(t+1) = a_{ji}(t) + \eta_a' \frac{b_{ji}(t)}{a_{ji}(t)} \delta_{ri} D^i. \tag{2.57}$$

Similarly,

$$
\begin{aligned}
b_{ji}(t+1) &= b_{ji}(t) - \eta_b \ln[(\frac{x_j - c_{ji}(t)}{a_{ji}(t)})^2]\delta_{ri}D^i, &(2.58)\\
c_{ji}(t+1) &= c_{ji}(t) + \eta_c' \frac{b_{ji}(t)}{x_j - c_{ji}(t)}\delta_{ri}D^i. &(2.59)
\end{aligned}
$$

where $\eta_c' = 2\eta_c$.

The tuning procedure similar to that for the gradient descent algorithm presented in Section 2.4.1 can be applied to tune a set of membership functions and to learn the optimized defuzzification parameters at the same time based on the modification formulae developed above.

2.5 Concluding Remarks

In this chapter, membership functions for a pattern recognition model have been discussed. In particular, three clustering techniques have been presented to generate the membership functions which can reflect the actual data distribution and therefore are more suitable for fuzzy pattern recognition applications. Among three clustering techniques discussed in the foregoing sections, the c-means algorithm is the simplest. However, the adaptive vector quantization (AVQ) algorithm provides a sophisticated estimation of the covariance matrix of a cluster by the learning process, but it does not guarantee that the learned covariance matrixes are positive-definite. If not, the method fails to generate useful membership functions. On the other hand, the self-organizing map (SOM), which takes into account the neighboring nodes in updating the connection weights, seems to be able to produce a more reliable solution than the other two techniques. The membership functions generated by these three techniques have been tested in various image and pattern recognition applications with good results. Experimental results on the generated membership functions for map image segmentation and handwritten digit recognition will be discussed in Chapter 6.

The heuristically chosen or generated membership functions from three clustering techniques can be tuned by using the gradient descent algorithm or a neural network approach for improved performance. The neural network method is actually based on the gradient descent algorithm. However, it provides us with a more sophisticated defuzzification by adopting a nonlinear transfer function in each of the output nodes. Moreover, the neural network approach combines tuning and optimization of defuzzification parameters together and therefore it is a more attractive technique.

Chapter 3

Optimal Image Thresholding

3.1 Introduction

In many image analysis applications, one usually needs to separate the foreground and background of an image to obtain useful information. For example, to analyze a document image, we need to extract characters and lines from the paper background. This is called image binarization and it can be carried out as a classification procedure to assign each pixel in the image to two classes, foreground (character and line pixels) and background.

Many pattern classification techniques can be used to separate the foreground and background of an image. The standard procedure is to extract a set of features at each pixel and then pass these features to a classifier to make a decision. Several supervised and unsupervised learning techniques described in Chapters 4 and 6 can be employed for this purpose. Using these methods, one can obtain a good performance by carefully selecting a set of robust input features and designing a reliable classifier. However, building a pixel classifier that can deal with many input features usually requires a large number of training samples and the resulting classification procedures are usually time consuming. In many practical applications, the amount of data to be processed can be very much larger. For example, in document image processing, one may need to process thousands of pages of text, thus a fast method is needed to extract characters from the paper background. The commonly used method is thresholding, that is, assigning a pixel to one class if its gray level is less than a specified threshold and otherwise assigning it to the other class [31], [32].

Thresholding is a simple pattern classification procedure in which there is only one input feature involved, the pixel intensity value. The key issue here is to choose an optimal threshold value so that the number of misclassified image pixels is kept as low as possible. Theoretically, we can determine the optimal threshold value according to the Bayes rule if we know the pixel value distributions of foreground and background classes [33], [34]. However, what we have in practice is a mixture of the two distributions, or the pixel value histogram of an image, but not the two separate distributions, so we may have to make some assumptions about the forms of the two distributions to simplify the problem. Under these assumptions, we can then determine the optimal threshold according to the statistical decision theory. Even so, threshold selection can still be complicated mathematically since it may involve the solution of nonlinear equations. Therefore, some further simplification or assumptions may be needed to find the optimal threshold.

Fuzzy system models can be used effectively to select the optimal threshold for an image. This is done by minimizing some measures of fuzziness. Several fuzzy quantities, such as fuzzy entropy, index of fuzziness and index of nonfuzziness (crispness) [35] can be used to describe the ambiguity in gray level of an ill-defined image for segmentation. To take fuzzy image geometry into account, Pal and Rosenfeld have developed the fuzzy compactness measure [35] and Pal and Ghosh have developed the index of area coverage [36] for image enhancement and thresholding. To find the optimal threshold based on the fuzzy compactness measure or the index of area coverage, the membership value of each pixel for all possible threshold values must be evaluated and be used to compute corresponding fuzzy quantity, thus these two methods require a long computing time.

Recently Huang and Wang have developed an image segmentation technique based on fuzzy entropy measure [37]. In their method, the image pixel membership functions are dependent of the threshold value and they reflect the distributions of pixel values in two classes, thus this method minimizes the classification error. We shall compare Huang and Wang's fuzzy thresholding algorithm with two of the best known threshold selection techniques, Otsu's method developed based on discriminant analysis [38] and Kittler and Illingworth's minimum error thresholding method [39]. Although these three methods have been developed based on different approaches, we shall prove that they can be derived under the same mathematical formulation and that the difference between the three methods is the choice of different weight functions for computing a criterion function.

In Section 3.2, we solve the threshold selection problem based on statistical decision theory. We assume that foreground and background pixels in an image follow two different probability distributions and our task is to fit the image histogram to the probability models. An iterative threshold selection method is described based on nonlinear optimization. In Section 3.3, we review Otsu's and Kittler and Illingworth's thresholding methods. In Section 3.4, Huang and Wang's fuzzy thresholding algorithm is described. In Section 3.5, we provide a unified description of all three methods. In Section 3.6, we describe the extension of the methods to multilevel thresholding. In Section 3.7, we apply the thresholding methods to real images. Our experimental results show that Huang and Wang's fuzzy thresholding method performs more reliably for different types of images when the contents of background and foreground in an image are well defined.

3.2 Threshold Selection Based on Statistical Decision Theory

Threshold selection can be formulated as a decision making problem. In this section, we first describe the optimal statistical decision rule for a two class problem, and apply this rule to Gaussian probability distributions. Then we solve the problem of threshold selection using the image pixel value histogram based on the results of statistical analysis.

3.2.1 Statistical Decision Rule

For simplicity, we consider a two class classification problem here. This corresponds to the most frequent requirement in image thresholding, to separate an image into two classes, foreground and background. Generalization of the formulation to multiclass problems is discussed in Section 3.6.

We make use of the following notations:

- $P_1 = \Pr\{C_1\}$ = *a priori* probability of class 1 denoted as C_1;
- $P_2 = \Pr\{C_2\}$ = *a priori* probability of class 2 denoted as C_2;
- the pixel intensity value is in the range $[0, L-1]$;
- $p_1(z) = \Pr\{z|C_1\}$ = probability density function of gray level z in class 1;
- $p_2(z) = \Pr\{z|C_2\}$ = probability density function of gray level z in class 2; and
- the probability density function of all pixel gray levels in the image, corresponding to the normalized pixel intensity histogram, is

$$p(z) = P_1 p_1(z) + P_2 p_2(z) \tag{3.1}$$

To make a classification of gray level z, we can simply evaluate the posterior probability of the two classes $\Pr\{C_1|z\}$ and $\Pr\{C_2|z\}$ and assign the gray level to the class with the larger posterior probability [34]. The Bayes decision rule for this two class problem is:

$$P_1 p_1(z) \underset{C_2}{\overset{C_1}{\gtrless}} P_2 p_2(z) \tag{3.2}$$

This equation means that a pixel having intensity z should be assigned to class 1 if the "$>$" condition is satisfied and to class 2 if the "$<$" condition is satisfied. If the two sides are equal, the assignment is arbitrary.

Let R_1 and R_2 denote the decision regions of C_1 and C_2 respectively, which means that Eq. (3.2) is satisfied for the ">" sign in region R_1 and for the "<" sign in region R_2, then the probability of error (the probability of a misclassification) is

$$E(R_1, R_2) = \sum_{z \in R_2} P_1 p_1(z) + \sum_{z \in R_1} P_2 p_2(z) \tag{3.3}$$

It can be proven that the probability of error is minimum when the decision rule in Eq. (3.2) is applied [34]. Thus, the decision rule is optimal. The optimal decision boundary can be determined from the following equation

$$P_1 p_1(z) = P_2 p_2(z) \tag{3.4}$$

In Eq. (3.3), the first summation gives the probability of error due to misclassification of class 1 samples to class 2, and the second summation the probability of error due to misclassification of class 2 samples to class 1. This is illustrated in Fig. 3.1. Obviously, the classification cannot be error free unless the two probability density functions do not overlap. In some applications, one can reduce this kind of overlap to reduce the recognition error by using more input features. In the thresholding problem, however, we only have one feature available, the pixel intensity, so the only option here is to choose an optimal threshold according to Eq. (3.2) to minimize the classification error.

For the case in Fig. 3.1, both R_1 and R_2 are continuous regions separated by a single threshold T. The thresholded image $g(x, y)$ of the input image $f(x, y)$ is defined as

$$g(x,y) = \begin{cases} 1 & \text{if } f(x,y) \leq T \\ 0 & \text{if } f(x,y) > T \end{cases} \tag{3.5}$$

All image pixels in $g(x, y)$ are labeled 0 or 1, corresponding to background and foreground respectively, assuming that the background is brighter than the foreground, such as in text document images. Note that the use of 0 and 1 here is purely arbitrary. One can use any other two different values to represent two classes. For example, 0 and 255 can be used to represent character and background pixels respectively for the display of the thresholded image on an 8-bit gray scale unit.

Two threshold values are needed for the case shown in Fig. 3.2. Since $P_2 p_2(z)$ decreases slowly and $P_1 p_1(z)$ decreases rapidly as z decreases to zero, $P_2 p_2(z)$ is ultimately larger than $P_1 p_1(z)$ for small z values. In this case, R_1 contains a single region $[T_1 + 1, T_2]$ and R_2 contains two separate regions $[0, T_1]$ and $[T_2 + 1, L - 1]$. The thresholded image $g(x, y)$ of the input image $f(x, y)$ is now defined as

$$g(x,y) = \begin{cases} 1 & \text{if } T_1 < f(x,y) \leq T_2 \\ 0 & \text{if } f(x,y) \leq T_1 \text{ or } f(x,y) > T_2 \end{cases} \tag{3.6}$$

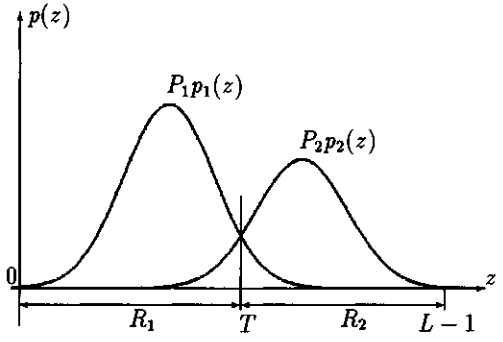


Figure 3.1 A two class decision problem. In this example, $R_1 = [0, T]$ and $R_2 = [T+1, L-1]$. The overlapping area corresponds to the probability of error. The error can be reduced by increasing the difference between the means of the two classes and decreasing the variance of each class.

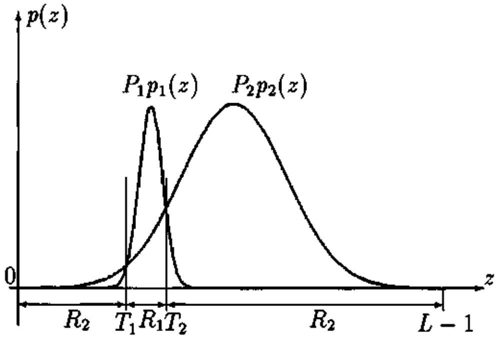


Figure 3.2 A two class problem that requires two thresholds.

3.2.2 Gaussian Distributions

If the two probability densities are Gaussian, then it is possible to obtain an analytical solution to Eq. (3.2). Let

$$p_1(z) = \frac{1}{\sqrt{2\pi}\sigma_1} \exp\left[-\frac{(z-m_1)^2}{2\sigma_1^2}\right] \tag{3.7}$$

and

$$p_2(z) = \frac{1}{\sqrt{2\pi}\sigma_2} \exp\left[-\frac{(z-m_2)^2}{2\sigma_2^2}\right] \tag{3.8}$$

Without loss of generality, we assume that

$$0 \le m_1 < m_2 \le L-1 \tag{3.9}$$

Substituting these equations into Eq. (3.2), we obtain

$$P_1 \frac{1}{\sqrt{2\pi}\sigma_1} \exp\left[-\frac{(z-m_1)^2}{2\sigma_1^2}\right] \underset{C_2}{\overset{C_1}{\gtrless}} \frac{1}{\sqrt{2\pi}\sigma_2} \exp\left[-\frac{(z-m_2)^2}{2\sigma_2^2}\right] P_2 \tag{3.10}$$

This equation can be simplified if we apply the logarithmic operation, ln(), on both sides since the ln() is a continuous and monotonically increasing function. The simplified result is

$$Az^2 + Bz + C \underset{C_2}{\overset{C_1}{\gtrless}} 0 \tag{3.11}$$

where

$$A = \sigma_1^2 - \sigma_2^2 \tag{3.12}$$
$$B = 2(m_1\sigma_2^2 - m_2\sigma_1^2) \tag{3.13}$$
$$C = \sigma_1^2 m_2^2 - \sigma_2^2 m_1^2 - 2\sigma_1^2\sigma_2^2 \ln(\frac{\sigma_2 P_1}{\sigma_1 P_2}) \tag{3.14}$$

The threshold value T can be determined by solving the following equation:

$$AT^2 + BT + C = 0 \tag{3.15}$$

This equation corresponds to Eq. (3.4) and gives the optimal decision boundary. In general, there are two threshold values:

$$T_1 = \frac{-B - \sqrt{B^2 - 4AC}}{2AC} \tag{3.16}$$

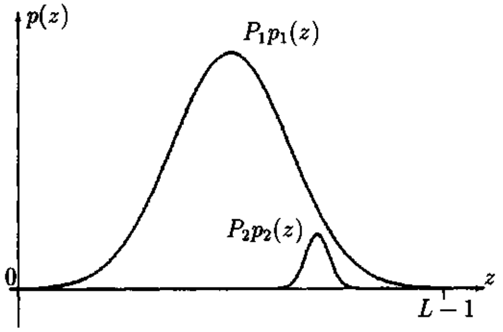


Figure 3.3 A two class problem for which Eq. (3.15) does not have a real valued solution and the optimal threshold cannot be determined based on the Bayes decision rule.

and

$$T_2 = \frac{-B + \sqrt{B^2 - 4AC}}{2AC} \tag{3.17}$$

Depending on the values of T_1 and T_2, the decision regions R_1 and R_2 can be determined as follows.

Case 1: $B^2 - 4AC < 0$. In this case, Eq. (3.15) does not have a real valued solution. This can happen if P_2 is very small so that that C is a very large positive number. This is illustrated in Fig. 3.3. In this case, the number of pixels in class 2 is less than the number of pixels in class 1 for any intensity values, thus there is no way to extract the pixels in class 2 based on intensity information alone.

Case 2: $T_1 \in [0, L-1]$, and $T_2 \notin [0, L-1]$. In this case, only T_1 is a useful solution and the decision regions are:

$$R_1 = [0, T_1],\ R_2 = [T_1 + 1, L-1] \tag{3.18}$$

In this equation, we arbitrarily assign pixels having gray level T_1 to class 1.

Case 3: $T_1 \notin [0, L-1]$, and $T_2 \in [0, L-1]$. In this case, only T_2 is a useful solution and the decision regions are:

$$R_1 = [0, T_2],\ R_2 = [T_2 + 1, L-1] \tag{3.19}$$

Case 4: $T_1 \in [0, L-1]$, and $T_2 \in [0, L-1]$. In this case, both T_1 and are useful solutions and the decision regions are:

$$R_1 = [T_1 + 1, T_2],\ R_2 = [0, T_1] \cup [T_2 + 1, L-1] \quad \text{if } T_1 < m_1 \tag{3.20}$$

and

$$R_1 = [0, T_1 - 1] \cup [T_2 + 1, L - 1], \; R_2 = [T_1 + 1, T_2] \qquad \text{if } T_1 > m_1 \tag{3.21}$$

Note that in this case one of the threshold values corresponding to the cross points of two Gaussian curves must be located between m_1 and m_2. Figure 3.2 shows an example of the decision regions in Eq. (3.20). If we exchange the variance values of the two densities in Fig. 3.2, we shall have the decision regions in a form shown in Eq. (3.21).

Case 5: $T_1 = T_2 \in [0, L-1]$. This happens when the two Gaussian probability densities have the same variance, that is, $\sigma_1 = \sigma_2 = \sigma$, or $A = 0$. In this case, the optimal threshold is

$$T = -\frac{C}{B} = \frac{m_1 + m_2}{2} + \frac{\sigma}{m_1 - m_2} \ln(\frac{P_2}{P_1}) \tag{3.22}$$

Furthermore, if the two *a priori* probabilities are the same, that is, $P_1 = P_2$, then

$$T = -\frac{C}{B} = \frac{m_1 + m_2}{2} \tag{3.23}$$

Image thresholding is ill-defined and cannot be carried out in case 1 without additional information as explained earlier. Although cases 2 and 3 are well defined theoretically, only the threshold that is between m_1 and m_2 is of practical interest in most real applications. For example, to deal with document images, we usually assume that background pixels are brighter than character pixels.

3.3 Non-fuzzy Thresholding Algorithms

3.3.1 Model Fitting Method

In Section 3.2, we have shown that it is possible to determine the optimal threshold value analytically if the the two class probability density functions are Gaussian with known parameters. In practical applications, however, the parameters in Eq. (3.10) are usually not known, so the formulae for calculation of optimal thresholds cannot be used directly. The only information we have is the mixture distribution, corresponding to $p(z)$ in Eq. (3.1), which can be obtained from the image pixel intensity histogram. To make use of the result in Section 3.2, we need to estimate the parameters of Gaussian distributions from the histogram data.

Let $h(z)$ be the normalized histogram function which represents the percentage of pixels having gray level z over the total number of pixels of the image. We can fit the Gaussian models in Eqs. (3.7) and (3.8) to the histogram function $h(z)$. That is, we can find a set of parameters so that the model probability density function $p(z)$ is as close to the actual probability density function $h(z)$ as possible. This can be carried out using a least squares errors (LSE) procedure. The mean square error between the two functions is

$$E(P_1, m_1, m_2, \sigma_1, \sigma_2) = \frac{1}{2}\sum_{z=0}^{L-1} \mid p(z) - h(z) \mid^2 \tag{3.24}$$

where $p(z)$ is given in Eq. (3.1). We need to determine parameters P_1, m_1, m_2, σ_1, and σ_2 so that the error function E is minimized.

Equation (3.24) is non-linear and does not have a known analytical solution. It can only be solved by numerical optimization, for example, using an iterative procedure based on the gradient information. For convenience, we represent the parameters using the following vector notation

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{bmatrix} = \begin{bmatrix} P_1 \\ m_1 \\ m_2 \\ \sigma_1 \\ \sigma_2 \end{bmatrix} \tag{3.25}$$

The derivative of E with respect to $\mathbf{v}$ is

$$\frac{\partial E}{\partial \mathbf{v}} = \begin{bmatrix} \partial E/\partial P_1 \\ \partial E/\partial m_1 \\ \partial E/\partial m_2 \\ \partial E/\partial \sigma_1 \\ \partial E/\partial \sigma_2 \end{bmatrix} = \sum_{z=0}^{L-1} [p(z) - h(z)] \begin{bmatrix} \partial p(z)/\partial P_1 \\ \partial p(z)/\partial m_1 \\ \partial p(z)/\partial m_2 \\ \partial p(z)/\partial \sigma_1 \\ \partial p(z)/\partial \sigma_3 \end{bmatrix} \tag{3.26}$$

where

$$\frac{\partial p(z)}{\partial P_1} = \frac{1}{\sqrt{2\pi}}\left\{\sigma_1^{-1}\exp\left[-\frac{(z-m_1)^2}{2\sigma_1^2}\right] - \sigma_2^{-1}\exp\left[-\frac{(z-m_2)^2}{2\sigma_2^2}\right]\right\} \tag{3.27}$$

$$\frac{\partial p(z)}{\partial m_1} = \frac{1}{\sqrt{2\pi}}P_1(z-m_1)\sigma_1^{-3}\exp\left[-\frac{(z-m_1)^2}{2\sigma_1^2}\right] \tag{3.28}$$

$$\frac{\partial p(z)}{\partial m_2} = \frac{1}{\sqrt{2\pi}}(1-P_1)(z-m_2)\sigma_2^{-3}\exp\left[-\frac{(z-m_2)^2}{2\sigma_2^2}\right] \tag{3.29}$$

$$\frac{\partial p(z)}{\partial \sigma_1} = \frac{1}{\sqrt{2\pi}}P_1\sigma_1^{-2}\left[\sigma_1^{-2}(z-m_1)^2 - 1\right]\exp\left[-\frac{(z-m_1)^2}{2\sigma_1^2}\right] \tag{3.30}$$

$$\frac{\partial p(z)}{\partial \sigma_2} = \frac{1}{\sqrt{2\pi}}(1-P_1)\sigma_2^{-2}\left[\sigma_2^{-2}(z-m_2)^2 - 1\right]\exp\left[-\frac{(z-m_2)^2}{2\sigma_2^2}\right] \tag{3.31}$$

Assuming that the value of $\mathbf{v}$ after iteration t is $\mathbf{v}^{(t)}$, we can improve the value in successive iterations based on the commonly used delta rule

$$\mathbf{v}^{(t+1)} = \mathbf{v}^{(t)} - \alpha\frac{\partial E}{\partial \mathbf{v}} \tag{3.32}$$

where α is a constant. In this equation, the value of $\mathbf{v}$ is moved along the opposite gradient direction to reduce the error function. The constant α should be sufficiently small to prevent overshooting.

Equation (3.32) represents a simplest method to solve a non-linear optimization problem. More sophisticated methods, such as the conjugate gradient algorithm [40] which also makes use of the gradient information, can be used for more efficient and robust computation.

Note that since the problem is nonlinear, the final solution strongly depends on the initial guess of the parameters $\mathbf{v}^{(0)}$. To set the initial values, we can make use of a non-iterative threshold selection method, such as one of the methods described in the following sections, and then calculate class *a priori* probabilities, means and variances. Assume that the threshold determined from a non-iterative method is T, then the initial parameter values can be computed as follows:

$$P_1^{(0)} = \sum_{z=0}^{T} h(z) \tag{3.33}$$

$$m_1^{(0)} = \frac{1}{P_1^{(0)}} \sum_{z=0}^{T} zh(z) \tag{3.34}$$

$$m_2^{(0)} = \frac{1}{1-P_1^{(0)}} \sum_{z=T+1}^{L-1} zh(z) \tag{3.35}$$

$$(\sigma_1^{(0)})^2 = \frac{1}{P_1^{(0)}} \sum_{z=0}^{T} (z-m_1^{(0)})^2 h(z) \tag{3.36}$$

$$(\sigma_2^{(0)})^2 = \frac{1}{1-P_1^{(0)}} \sum_{z=T+1}^{L-1} (z-m_2^{(0)})^2 h(z) \tag{3.37}$$

Once the parameters are obtained, we can then calculate the optimal threshold value according to the formulae described in Section 3.2.

3.3.2 Otsu's Thresholding Method

Otsu has developed a thresholding method based on discriminant analysis which maximizes some measures of class separability [38]. One of the measures is

$$J_{OT_1}(T) = \frac{P_1(T)P_2(T)\,[m_1(T)-m_2(T)]^2}{P_1(T)\sigma_1^2(T)+P_2(T)\sigma_2^2(T)} \tag{3.38}$$

where

$$P_1(T) = \Pr\{C_1\} = \sum_{z=0}^{T} h(z) \tag{3.39}$$

$$P_2(T) = \Pr\{C_2\} = \sum_{z=T+1}^{L-1} h(z) = 1 - P_1 \tag{3.40}$$

$$m_1(T) = \sum_{z=0}^{T} z\Pr\{z|C_1\} = \frac{1}{P_1}\sum_{z=0}^{T} zh(z) \tag{3.41}$$

$$m_2(T) = \sum_{z=T+1}^{L-1} z\Pr\{z|C_2\} = \frac{1}{P_2}\sum_{z=T+1}^{L-1} zh(z) \tag{3.42}$$

$$\sigma_1^2(T) = \sum_{z=0}^{T}[z - m_1(T)]^2\Pr\{z|C_1\} = \frac{1}{P_1}\sum_{z=0}^{T}[z - m_1(T)]^2 h(z) \tag{3.43}$$

$$\sigma_2^2(T) = \sum_{z=T+1}^{L-1}[z - m_2(T)]^2\Pr\{z|C_2\} = \frac{1}{P_2}\sum_{z=T+1}^{L-1}[z - m_2(T)]^2 h(z) \tag{3.44}$$

In the above equations, C_1 and C_2 are dependent of T and contain pixels with gray values in $[0, T]$ and $[T+1, L-1]$ respectively.

To maximize the criterion function in Eq. (3.38), the means of the two classes should be as well separated as possible and the variances in both classes should be as small as possible. This is similar to the Fisher criterion for pattern classification [34].

The optimal threshold value T_{OT}^* can be determined by searching for the value in the range $[0, L-1]$ so that $J_{OT}(T)$ is the maximum. That is,

$$T_{OT}^* = \arg\max_{0\le T\le L-1} J_{OT_1}(T) \tag{3.45}$$

Ostu has pointed out that the criterion function $J_{OT}(T)$ is equivalent to the following two alternative functions [38]

$$J_{OT_2}(T) = \frac{\sigma^2}{P_1(T)\sigma_1^2(T) + P_2(T)\sigma_2^2(T)} \tag{3.46}$$

and

$$J_{OT_3}(T) = \frac{P_1(T)P_2(T)\,[m_1(T) - m_2(T)]^2}{\sigma^2} \tag{3.47}$$

where

$$\sigma^2 = \sum_{z=0}^{L-1}[z - m(T)]^2 h(z) \tag{3.48}$$

where

$$m = \sum_{z=0}^{L-1} zh(z) = P_1 m_1(T) + P_2 m_2(T) \tag{3.49}$$

3.3.3 Minimum Error Thresholding Method

Kittler and Illingworth have used the following criterion function to select the optimal threshold for a given histogram function $h(z)$ [39]

$$J_{KI}(T) = \sum_{z=0}^{L-1} h(z)c_{KI}(z,T) = \sum_{z=0}^{T} h(z)c_{KI}^{(1)}(z,T) + \sum_{z=T+1}^{L-1} h(z)c_{KI}^{(2)}(z,T) \tag{3.50}$$

where $c_{KI}^{(1)}(z,T)$ and $c_{KI}^{(2)}(z,T)$ can be considered as being two cost functions for pixels in two classes and

$$c_{KI}(z,T) = \begin{cases} c_{KI}^{(1)}(z,T) & \text{if } z \le T \\ c_{KI}^{(2)}(z,T) & \text{if } z > T \end{cases} \tag{3.51}$$

We shall show in Section 3.5 that the function in Eq. (3.50) can also be used to formulate both Otsu's thresholding method described in Section 3.3.2 and the fuzzy thresholding method to be discussed in Section 3.4. Thus, Eq. (3.50) can be considered as being a general criterion function for deriving a class of optimal thresholding selection algorithms. In fact, we can view $c_{KI}(z,T)$ as a cost for pixels with gray level z and $J_{KI}(T)$ as the average cost for all pixels in the image to binarize the image at threshold T.

In Kittler and Illingworth's minimum error thresholding method, the cost function $c_{KI}(z,T)$ is derived based on the Bayes rule. Suppose that pixels in the two classes separated by threshold T have probability density functions $p_1(z|C_1,T)$ and $p_2(z|C_2,T)$ respectively. The class conditional probability of a gray level z being assigned to a class correctly is [39]

$$\Pr\{z,T\} = \begin{cases} \Pr\{C_1|z,T\} = \dfrac{P_1(T)}{h(z)} p_1(z|C_1,T) & \text{if } z \le T \\ \Pr\{C_1|z,T\} = \dfrac{P_2(T)}{h(z)} p_2(z|C_2,T) & \text{if } z > T \end{cases} \tag{3.52}$$

Assuming that $p_1(z|C_1,T)$ and $p_2(z|C_2,T)$ are Gaussian with parameters shown in Eqs. (3.7) to (3.8), we have

$$\Pr\{z,T\} = \begin{cases} \dfrac{P_1(T)}{h(z)} \exp\left[-\dfrac{(z-m_1(T))^2}{2\sigma_1(T)^2}\right] & \text{if } z \le T \\ \dfrac{P_2(T)}{h(z)} \exp\left[-\dfrac{(z-m_2(T))^2}{2\sigma_2(T)^2}\right] & \text{if } z > T \end{cases} \tag{3.53}$$

Ignoring $h(z)$ since it is independent of the class and threshold value and taking the logarithm of the function, we obtain the following functions which can be used as the cost functions in Eq. (3.50)

$$c_{KI}(z,T) = \begin{cases} c_{KI}^{(1)}(z,T) = \dfrac{[z - m_1(T)]^2}{\sigma_1^2(T)} \\ \qquad +2\ln\sigma_1(T) - 2\ln P_1(T) \quad \text{if } z \le T \\ c_{KI}^{(2)}(z,T) = \dfrac{[z - m_2(T)]^2}{\sigma_2^2(T)} \\ \qquad +2\ln\sigma_2(T) - 2\ln P_2(T) \quad \text{if } z > T \end{cases} \tag{3.54}$$

Combining Eq. (3.50) to Eq. (3.54), we obtain

$$J_{KI}(T) = 1 + 2[P_1(T)\ln\sigma_1(T) + P_2(T)\ln\sigma_2(T)] \tag{3.55}$$

$$- 2[P_1(T)\ln P_1(T) + P_2(T)\ln P_2(T)] \tag{3.56}$$

The optimal threshold is

$$T_{KI}^* = \arg\max_{0 \le T \le L-1} J_{KI}(T) \tag{3.57}$$

3.4 Fuzzy Thresholding Algorithm

Fuzzy logic is a powerful tool to solve many image processing problems because of its ability to deal with ambiguous data. Selection of a threshold to binarize an image is usually not straight forward because of ambiguity or fuzziness caused by the overlapping of the two class probability densities. Several fuzzy model based methods have been developed in the past to overcome the difficulties. For example, an optimal threshold value can be determined based on Pal and Rosenfeld's fuzzy compactness measure and Pal and Ghosh's index of area coverage measure [35], [36]. In these two methods, for each possible threshold value in the range $[0, L-1]$, the membership value of every pixel is needed to compute the compactness or index of area coverage measure. When the image is large in size, these methods can require a long computing time to search for the optimal threshold value. Huang and Wang have recently developed a fuzzy thresholding algorithm which makes use of the image pixel value histogram but does not need deal with each individual pixel [37]. This method can be very efficiently implemented. In this section, we describe Huang and Wang's method and in Section 3.5, we show that this method can be formulated with a criterion function similar to those used in Otsu's and the minimum error thresholding methods.

3.4.1 Fuzzy Membership Functions

We consider an image as an array of fuzzy singletons corresponding to image pixels, each having a membership value associated with a certain property of the pixel [41], [35]. Under this assumption, an image I can be represented as

$$I = \{(f(x,y), \mu_I(f(x,y)))\} \tag{3.58}$$

For image thresholding, the membership function $\mu_I(f(x,y))$ can be defined in terms of the grade of pixel (x,y) belonging to one of the two classes, background and foreground.

In Huang and Wang's method, the membership function is defined as

$$\mu_I(f(x,y)) = \begin{cases} \dfrac{1}{1+|f(x,y)-m_1(T)|/D} & \text{if } f(x,y) \le T \\ \dfrac{1}{1+|f(x,y)-m_2(T)|/D} & \text{if } f(x,y) > T \end{cases} \tag{3.59}$$

where m_1 and m_2 are the averages of the gray levels of the pixels in two classes respectively and are given by

$$m_1(T) = \sum_{z=0}^{T} zh(z) / \sum_{z=0}^{T} h(z) \tag{3.60}$$

and

$$m_2(T) = \sum_{z=T+1}^{L-1} zh(z) / \sum_{z=T+1}^{L-1} h(z) \tag{3.61}$$

and D is a constant chosen in such a way that

$$0.5 \le \mu_I(f(x,y)) \le 1 \tag{3.62}$$

The reason for restricting the membership values in the range $[0.5, 1.0]$ is to make use of the entropy measure described in Section 3.4.2. Huang and Wang choose

$$D = z_{\text{max}} - z_{\text{min}} \tag{3.63}$$

where z_{min} and z_{max} are minimum and maximum gray levels of the image respectively. Note that z_{min} and z_{max} may not necessary be 0 and $L-1$ respectively. For the choice of D in Eq. (3.63), the condition in Eq. (3.62) is satisfied.

The shape of the membership function is shown in Fig. 3.4 for $m_1 = 80$, $m_2 = 170$, $D = 256$ and $T = 120$. In each of the two classes, the membership value is the largest (equal to 1) at the class average gray level (m_1 or m_2) and reduces when the difference between the pixel gray level and its class average gray level increases. This means that pixels with gray levels close to their corresponding class average gray levels have less fuzziness or ambiguity and thus can be classified with a larger confidence than pixels with gray levels far from their class average gray levels.

3.4.2 Threshold Selection Based on Fuzzy Measure

Huang and Wang have made use of the entropy measure as the criterion function for selection of the optimal image threshold [37]. It is defined as

$$J_{HW}(T) = E(I) = -\frac{1}{MN\ln 2} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} S_e(\mu_I(f(x,y))) \tag{3.64}$$

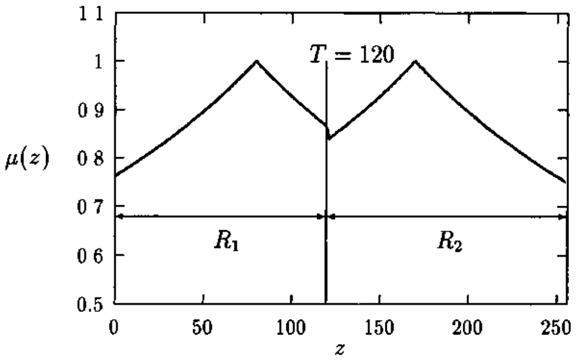


Figure 3.4 Membership function in Eq. (3.59) for $m_1 = 80$, $m_2 = 170$, $D = 256$ and $T = 120$.

where

$$S_e(\mu) = -\mu \ln \mu - (1-\mu)\ln(1-\mu) \quad (0 \le \mu \le 1) \tag{3.65}$$

is the Shannon's entropy function shown in Fig. 3.5. This function is used here as a cost function. Since the cost should decrease as the membership value increases (as the fuzziness becomes smaller), we can only make use of the Shannon function for the interval $0.5 \le \mu \le 1$. This is why we need to impose the condition in Eq. (3.62). If we group all pixels having the same gray levels in Eq. (3.64), then the equation can be simplified using the histogram information $h(z)$ as follows

$$J_{HW}(T) = E(I) = -\frac{1}{\ln 2} \sum_{z=0}^{L-1} h(z) S_e(\mu_I(z)) \tag{3.66}$$

The optimal threshold is chosen to minimize $J_{HW}(T)$, that is,

$$T^*_{HW} = \arg \min_{0 \le T \le L-1} J_{HW}(T) \tag{3.67}$$

The entropy measure has the following properties [37]:

(1) $J_{HW}(T) = E(I)$ is large if many pixels have membership values close to 0.5 or their gray levels are far from their class average gray levels. It has the maximum value 1 if all membership values equal to 0.5.

(2) $J_{HW}(T) = E(I)$ is small if many pixels have membership values close to ~~0.5~~ 1.0 or their gray levels are close to their class average gray levels. It has the minimum value 0 if all membership values are equal to 1.

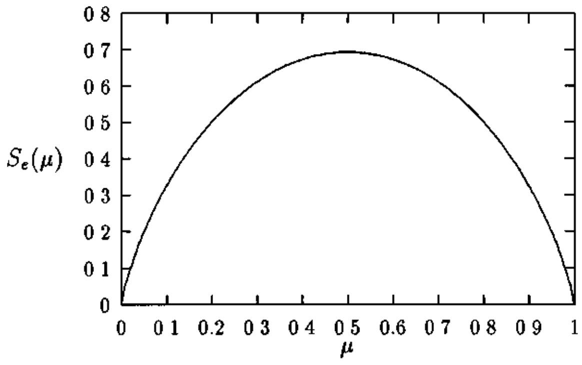


Figure 3.5 Shannon's entropy function.

(3) $E(I) \leq E(I')$ if I is crisper (less fuzzy) than I'. In this case, I have pixel gray levels distributed more compactly around the two class average gray levels than I'.

From the above properties, we can see that the optimal threshold obtained based on the entropy criterion corresponds to an assignment of the image pixels to two classes in a way such that the gray levels of the pixels in the two classes are as close to their class average gray levels as possible. This makes the two class means as well separated as possible and the two class variances as small as possible. Therefore, this criterion is consistent with Otsu's method [38] and the Fisher criterion [34]. A further comparison of three non-iterative thresholding methods are described in Section 3.5.

3.4.3 Fast Computation

The class average gray levels can be computed efficiently using a recursive method [37]. We can make use of the following functions

$$S(T) = \sum_{z=0}^{T} h(z), \quad \bar{S}(T) = \sum_{z=T+1}^{L-1} h(z) \tag{3.68}$$

$$W(T) = \sum z = 0^T zh(z), \quad \bar{W}(T) = \sum_{z=T+1}^{L-1} zh(z) \tag{3.69}$$

The algorithm has the following steps:

(1) Determine z_{min} and z_{max}, the minimum and the maximum gray levels in the image.

(2) Set $\bar{S}(L-1)=0$ and $\bar{W}(0)=0$.

(3) Set $T=z_{\text{min}}+1$, $J^*_{HW}=1$ (the maximum possible value), and $T^*_{HW}=T$.

(4) Compute

$$\begin{aligned} S(T) &= S(T-1)+h(T), \quad \bar{S}(T)=S(L-1)-S(T) && (3.70)\\ W(T) &= S(T-1)+Th(T), \quad \bar{W}(T)=W(L-1)-W(T) && (3.71)\\ m_1(T) &= W(T)/S(T), \quad m_2(T)=\bar{W}(T)/\bar{S}(T) && (3.72)\end{aligned}$$

(5) Compute $\mu_I(z)$ for $0\le z\le L-1$ and $J_{HW}(T)$.

(6) If $J^*_{HW}>J_{HW}(T)$, then set $J^*_{HW}=J_{HW}(T)$ and $T^*_{HW}=T$.

(7) Set $T=T+1$. If $T<z_{\text{max}}$, go to step (4).

When the algorithm terminates, T^*_{HW} contains the optimal threshold value.

3.5 Unified Formulation of Three Thresholding Algorithms

3.5.1 Analytical Formulation

In this section, we show that all three non-iterative thresholding algorithms, Otsu's method, Kitller and Illingworth's minimum error based method, and Huang and Wang's fuzzy model based method, can be formulated using the following unified equation:

$$J(T)=\sum_{z=0}^{L-1}h(z)c(z,T)=\sum_{z=0}^{T}h(z)c_1(z,T)+\sum_{z=T+1}^{L-1}h(z)c_2(z,T) \qquad (3.73)$$

where $c(z,T)$ can be considered as the cost to pixels with gray level z when the threshold is set at value T. The cost function is split into two parts, $c_1(z,T)$ and $c_2(z,T)$, which provide different weights for pixels in two classes.

For Otsu's method, we make use of the following criterion function

$$J_{OT}(T)=\frac{P_1(T)\sigma_1^2(T)+P_2(T)\sigma_2^2(T)}{\sigma^2} \qquad (3.74)$$

This is simply the inverse of $J_{OT_2}(T)$. Now the optimal threshold is the gray level at which the criterion function $J_{OT}(T)$ is minimum, similar to the other two criterion functions. That is,

$$T^*_{OT}=\arg\min_{0\le T\le L-1}J_{OT}(T) \qquad (3.75)$$

Based on the definitions of $P_1(T), P_2(T), \sigma_1(T), \sigma_2(T)$ and σ in Eqs. (3.39) to (3.44) and (3.48) to (3.49), we can rewrite the modified criterion function as

$$J_{OT}(T) = \sum_{z=0}^{T} h(z)\frac{[z-m_1(T)]^2}{\sigma^2} + \sum_{z=T+1}^{L-1} h(z)\frac{[z-m_2(T)]^2}{\sigma^2} \tag{3.76}$$

Thus, the cost functions are

$$c_{OT}(z,T) = \begin{cases} c_{OT}^{(1)}(z,T) = \dfrac{[z-m_1(T)]^2}{\sigma^2} & \text{if } z \le T \\ c_{OT}^{(2)}(z,T) = \dfrac{[z-m_2(T)]^2}{\sigma^2} & \text{if } z > T \end{cases} \tag{3.77}$$

The minimum error thresholding method was developed based on the criterion function shown in Eq. (3.50) and the cost functions are given in Eq. (3.54).

For Huang and Wang's fuzzy thresholding method, substituting Eqs. (3.59) and (3.65) into Eq. (3.66), we get

$$\begin{aligned} J_{HW}(T) &= \frac{1}{\ln 2}\sum_{z=0}^{T} h(z)\left\{\ln\left[1+\frac{z-m_1(T)}{D}\right] + \frac{z-m_1(T)}{D+z-m_1(T)}\ln\frac{z-m_1(T)}{D}\right\} \\ &\quad + \frac{1}{\ln 2}\sum_{z=T+1}^{L-1} h(z)\left\{\ln\left[1+\frac{z-m_2(T)}{D}\right]\right. \\ &\quad \left. + \frac{z-m_2(T)}{D+z-m_2(T)}\ln\frac{z-m_2(T)}{D}\right\} \end{aligned} \tag{3.78}$$

Thus, the cost functions are

$$c_{HW}(z,T) = \begin{cases} c_{HW}^{(1)}(z,T) = \dfrac{1}{\ln 2}\left\{\ln\left[1+\dfrac{z-m_1(T)}{D}\right]\right. \\ \quad \left. + \dfrac{z-m_1(T)}{D+z-m_1(T)}\ln\dfrac{z-m_1(T)}{D}\right\} & \text{if } z \le T \\ c_{HW}^{(2)}(z,T) = \dfrac{1}{\ln 2}\left\{\ln\left[1+\dfrac{z-m_2(T)}{D}\right]\right. \\ \quad \left. + \dfrac{z-m_2(T)}{D+z-m_2(T)}\ln\dfrac{z-m_1(T)}{D}\right\} & \text{if } z > T \end{cases} \tag{3.79}$$

3.5.2 Analysis of Cost Functions

From Eqs. (3.77), (3.54) and (3.79), we can see now that the difference between the three non-iterative methods is the choice of the cost functions. Otsu's method makes use of two quadratic functions as cost functions, whose centers are at the class average gray levels. Despite the difference in center positions, the parabola corresponding to each cost function has the same overall shape for different T values. The minimum error thresholding method also makes use of two quadratic functions as cost functions and their centers are also located at the class average gray levels. In this case, however, the parameters of the functions are dependent of T. During a search for the optimal threshold value, as T varies from $z_{\min}$ to $z_{\max}$, the cost functions also change. Huang and Wang's fuzzy thresholding method is similar to Otsu's method in the sense that two cost functions also have fixed parameters (independent of T). The difference between the two methods is that different forms of cost functions are used.

The difference between the cost functions used in the three non-iterative methods can be further analyzed with numerical simulation. For the histogram shown in Fig. 3.6, the cost functions for three T values, 100, 170 and 210, are shown in Figs. 3.7 to 9. We can see that the cost functions are minimum at class average gray level values. As z moves away from its corresponding class average gray level, the cost function increases. All cost functions used in the three methods share these common characteristics.

For each T value, the cost function used in Otsu's method consists of two parabolic segments. The parabolas have the same shape, but the segments are taken in different intervals. The minimum values of the cost function in two regions separated by threshold T, corresponding to the bottom points of the parabolas, are always zero, which occurs at the class average gray levels.

For each T, the cost function used in the minimum error thresholding method also consists of two parabolic segments. However, for different T values, the parabolas have different shapes. As T increases, the variance of class 1 increases and the corresponding parabola becomes wider. At the same time, the variance of class 2 decreases and the corresponding parabola becomes narrower. For different T values, the minimum values of the cost function in two regions, corresponding to the bottom points of the parabolas, are also different due to the log terms in Eq. (3.54).

For each T value, the cost function used in Huang and Wang's fuzzy thresholding method consists of two wedge-shaped segments. The corresponding curves have the same characteristics as the parabolas in Otsu's method.

In Huang and Wang's fuzzy thresholding method, the cost function increases more rapidly as the gray level moves away from the class average gray level in each region, so the criterion function may be more discriminative against gray levels distant from their class means. Although it is still an open problem how to design an optimal cost function, our experimental results in Section 3.7 do show that Huang and Wang's method has an overall better performance than other methods.

3.6 Multilevel Thresholding

In Sections 3.2 to 3.5, we have only considered two-class classification problems. That is, a pixel is assigned to either foreground or background. In many applications,

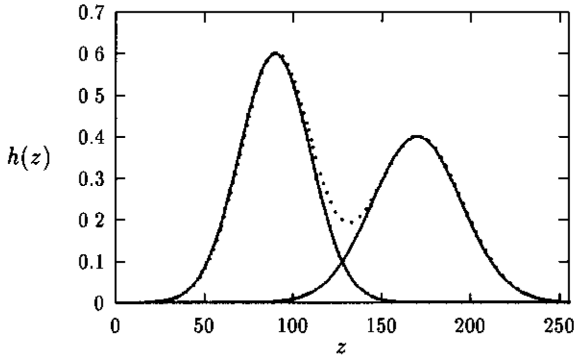


Figure 3.6 A histogram for analyzing the difference between the cost functions used in three non-iterative methods. The histogram (dotted line) contains two Gaussian probability densities (solid lines).

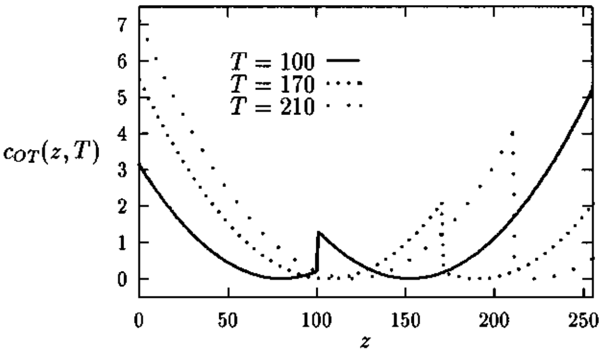


Figure 3.7 The cost function used in Otsu's method for the histogram shown in Fig. 3.6 for three T values.

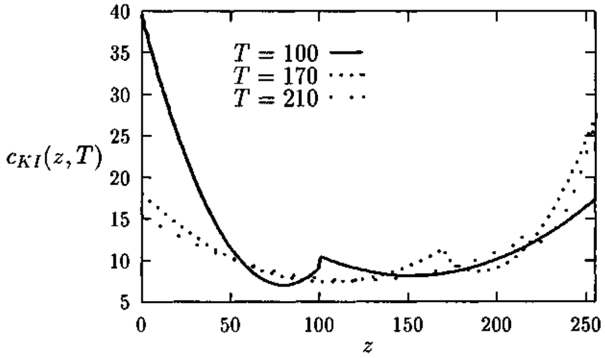


Figure 3.8 The cost function used in the minimum error thresholding method for the histogram shown in Fig. 3.6 for three T values.

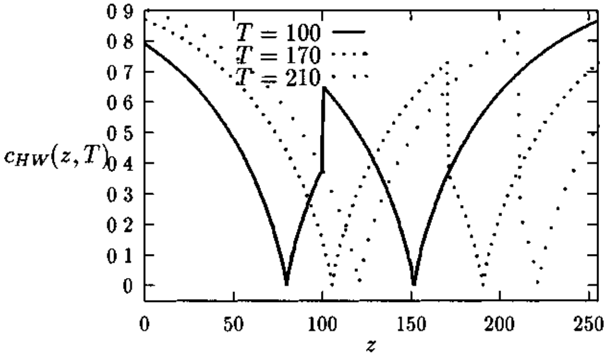


Figure 3.9 The cost function used in Huang and Wang's fuzzy thresholding method for the histogram shown in Fig. 3.6 for three T values.

multilevel thresholding may be needed to segment an image into more than two classes [39]. Assume that there are K classes, $C_1, C_2, \cdots, C_K$, in the image and that our task is to determine K optimal decision regions, $R_1, R_2, \cdots, R_K$, to segment the image. All algorithms described in Sections 3.2 to 3.5 can be generalized to deal with the multilevel thresholding problem.

3.6.1 Generalization of Model Fitting Method

To extend the model fitting method to multilevel thresholding, we assume that the histogram of the image is generated by the following model:

$$p(z) = \sum_{k=1}^{K} p_k(z) = \sum_{k=1}^{K} \frac{P_k}{\sqrt{2\pi}\sigma_k} \exp\left[-\frac{(z-m_k)^2}{2\sigma_k^2}\right] \tag{3.80}$$

The mean square error criterion function is now

$$E(P_1, \cdots, P_{K-1}, m_1, \cdots, m_K, \sigma_1, \cdots, \sigma_K) = \frac{1}{2}\sum_{z=0}^{L-1} \mid p(z) - h(z) \mid^2 \tag{3.81}$$

and the parameter vector is

$$\mathrm{v} = [P_1 \ \cdots \ P_{K-1} \ m_1 \ \cdots \ m_K \ \sigma_1 \ \cdots \ \sigma_K]^T \tag{3.82}$$

where superscript T means the matrix transpose operation. Note that there are only $K-1$ independent class *a priori* probabilities since

$$P_1 + P_2 + \cdots + P_K = 1 \tag{3.83}$$

Thus, Eq. (3.1) can be rewritten as

$$\begin{aligned} p(z) &= \sum_{k=1}^{K-1} \frac{p_k}{\sqrt{2\pi}\sigma_k} \exp\left[-\frac{(z-m_k)^2}{2\sigma_k^2}\right] \\ &\quad + \frac{1-(P_1+P_2+\cdots+P_{K-1})}{\sqrt{2\pi}\sigma_K} \exp\left[-\frac{(z-m_K)^2}{2\sigma_K^2}\right] \end{aligned} \tag{3.84}$$

By changing 1 to k ($k = 1, 2, \cdots, K-1$), 2 to K and $1-P_1$ to $1-P_1-P_2-\cdots-P_{K-1}$, we can make use of Eqs. (3.27) to (3.31) to calculate the derivatives of E with respect to the variables. The delta rule in Eq. (3.32) can still be used to find the optimal solution of v by minimizing E. Again using one of the non-iterative methods to be described below, we can choose a set of initial threshold values and estimate the initial parameter values for use in Eq. (3.32) as

$$P_k^{(0)} = \sum_{z=T_{k-1}}^{T_k} h(z) \tag{3.85}$$

$$m_k^{(0)} = \frac{1}{P_k^{(0)}} \sum_{z=0}^{T} z h(z) \tag{3.86}$$

$$(\sigma_k^{(0)})^2 = \frac{1}{P_k^{(0)}} \sum_{z=0}^{T} (z - m_k^{(0)})^2 h(z) \tag{3.87}$$

In theory, we should assign a pixel with gray level z to class class(z) such that

$$\text{class}(z) = \arg \max_{1 \le k \le K} P_k p_k(z) \tag{3.88}$$

In practice, however, this can lead to very complicated decision regions. To simplify the problem, we assume that the threshold between two neighboring classes is only related to the probability density functions of the two classes and that the threshold is located between the means of the two classes. Under these assumptions, we only need to select $K-1$ thresholds $T_1, T_2, \cdots, T_{K-1}$ which satisfy the following condition

$$m_1 \le T_1 \le m_2 \le T_2 \le \cdots m_{K-1} \le T_{K-1} \le m_K \tag{3.89}$$

The decision rule can now be simplified as

$$\text{class}(z) = \begin{cases} C_1 & \text{if } z < m_1 \\ \arg \max_{k=j,j+1} P_k p_k(z) & \text{if } m_1 \le z < m_K \\ C_K & \text{if } z > m_K \end{cases} \tag{3.90}$$

where j ($j = 1, 2, \cdots, K-1$) is chosen such that

$$m_j \le z < m_{j+1}$$

Threshold value T_k ($k = 1, 2, \cdots, K-1$) can be determined from the roots of the following quadratic equation:

$$A_k T_k^2 + B T_k + C_k = 0 \tag{3.91}$$

where

$$A_k = \sigma_{k-1}^2 - \sigma_k^2 \tag{3.92}$$

$$B_k = 2(m_{k-1}\sigma_k^2 - m_k \sigma_{k-1}^2) \tag{3.93}$$

$$C_k = \sigma_{k-1}^2 m_k^2 - \sigma_k^2 m_{k-1}^2 - 2\sigma_{k-1}^2 \sigma_k^2 \ln\left(\frac{\sigma_k P_{k-1}}{\sigma_{k-1} P_k}\right) \tag{3.94}$$

3.6.2 Generalization of Non-iterative Methods

Our task is to select K optimal threshold values $T_1, T_2, \cdots, T_{K-1}$ which satisfy the following condition

$$T_1 < T_2 < \cdots < T_{K-1} \tag{3.95}$$

For convenience to describe class parameters, we add two fixed thresholds below:

$$T_0 = -1, \;\; T_K = L - 1 \tag{3.96}$$

We make use of the following parameters estimated from the histogram data to characterize the pixel value distribution in class k $(k = 1, \cdots, K)$.
Class *a priori* probabilities:

$$P_k(T_1, \cdots, T_{K-1}) = \sum_{z=T_{k-1}}^{T_k} h(z) \tag{3.97}$$

Note that there are only $K - 1$ independent class *a priori* probabilities since

$$\sum_{k=1}^{K} P_k(T_1, T_2, \cdots, T_{K-1}) = 1 \tag{3.98}$$

Class means:

$$m_k(T_1, \cdots, T_{K-1}) = \frac{1}{P_k(T_1, T_2, \cdots, T_{K-1})} \sum_{z=T_{k-1}}^{T_k - 1} z h(z) \tag{3.99}$$

Class variances:

$$\sigma_k^2(T_1, \cdots, T_{K-1}) = \frac{1}{P_k(T_1, T_2, \cdots, T_{K-1})} \sum_{z=T_{k-1}}^{T_k - 1} [z - m_k(T_1, \cdots, T_{K-1})]^2 h(z) \tag{3.100}$$

Total mean:

$$m = \sum_{z=0}^{L-1} z h(z) = \sum_{k=1}^{K} P_k(T_1, T_2, \cdots, T_{K-1}) m_k(T_1, \cdots, T_{K-1}) \tag{3.101}$$

Total variance:

$$\sigma^2 = \sum_{z=0}^{L-1} (z - m)^2 h(z) \tag{3.102}$$

We can determine the optimal thresholds using the following criterion functions:

$$\begin{aligned}
J_{OT}(T_1, \cdots, T_{K-1}) &= \sum_{k=1}^{K-1} \sum_{z=T_{k-1}}^{T_k - 1} h(z) c_{OT}^{(k)}(T_1, \cdots, T_{K-1}) & (3.103) \\
J_{KI}(T_1, \cdots, T_{K-1}) &= \sum_{k=1}^{K-1} \sum_{z=T_{k-1}}^{T_k - 1} h(z) c_{KI}^{(k)}(T_1, \cdots, T_{K-1}) & (3.104) \\
J_{HW}(T_1, \cdots, T_{K-1}) &= \sum_{k=1}^{K-1} \sum_{z=T_{k-1}}^{T_k - 1} h(z) c_{HW}^{(k)}(T_1, \cdots, T_{K-1}) & (3.105)
\end{aligned}$$

where in the interval $T_{k-1} \leq z < T_k$ $(0 \leq k \leq K-1)$ the cost functions are

$$c_{OT}^{(k)}(z, T_1, \cdots, T_{K-1}) = \frac{[z - m_k(T_1, \cdots, T_{K-1})]^2}{\sigma^2} \tag{3.106}$$

$$\begin{aligned} c_{KI}^{(k)}(z, T_1, \cdots, T_{K-1}) = {} & \frac{[z - m_k(T_1, \cdots, T_{K-1})]^2}{\sigma_k^2(T_1, \cdots, T_{K-1})} \\ & +2 \ln \sigma_k(T_1, \cdots, T_{K-1}) \\ & -2 \ln P_k(T_1, \cdots, T_{K-1}) \end{aligned} \tag{3.107}$$

$$\begin{aligned} c_{HW}^{(k)}(z, T_1, \cdots, T_{K-1}) = {} & \frac{1}{\ln 2} \left\{ \ln \left[1 + \frac{z - m_k(T_1, \cdots, T_{K-1})}{D} \right] \right. \\ & + \frac{z - m_k(T_1, \cdots, T_{K-1})}{D + z - m_k(T_1, \cdots, T_{K-1})} \\ & \left. \ln \frac{z - m_k(T_1, \cdots, T_{K-1})}{D} \right\} \end{aligned} \tag{3.108}$$

3.7 Applications

In this section, we apply the optimal threshold selection methods described earlier to real images (see Fig. 3.10 to Fig. 3.17) for background and foreground separation. These images were scanned into the computer as 8-bit gray scale pictures from printed material using a 300 dpi Microtek scanner. For each image, we compute five thresholds based on Otsu's method, model fitting method 1, Kittler and Illingworth's minimum error method, model fitting method 2, and Huang and Wang's fuzzy thresholding method. In model fitting methods 1 and 2, we make use of the threshold values obtained from Otsu's method and Kittler and Illingworth's method respectively and then estimate the initial parameters according to Eqs. (3.33) to (3.37) and compute the optimal threshold values according to Eqs. (3.27) to (3.32).

3.7.1 Segmentation of Document Images

Figures 3.10 to 3.13 show four test images "poster 1", "poster 2", "label" and "face" and their gray level histograms. Our task here is to extract characters and line drawings from the background. The threshold values determined using the five methods are shown in Table 3.1 and the thresholded images are shown in Figs. 3.18 to 3.21.

The two model fitting methods make use of the same algorithm to solve the nonlinear optimization problem, but start with different initial solutions. Theoretically, the model fitting algorithm should produce the optimal result if the class probability densities are Gaussian. In practice, however, because the probability densities may not be exactly Gaussian for a real image and because the problem is nonlinear, the final results are strongly dependent of the initial estimate of the Gaussian parameters. The thresholding results obtained with model fitting method 1 are close to those obtained with Otsu's method and results obtained with model fitting method 2 are close

Table 3.1 Optimal threshold values determined by five thresholding methods for four document images.

Image	Thresholding Method				
	Otsu	Model Fitting 1	KI Min Error	Model Fitting 2	HW Fuzzy
Poster 1	95	85	52	47	56
Poster 2	117	114	95	95	108
Label	195	199	186	191	202
Face	114	106	73	68	85

to those obtained with the minimum error method. For "poster 1", the thresholded image obtained with model fitting method 1 does show some improvement over that obtained with Otsu's method.

"Poster 1" and "poster 2" have textured background and "label" has a poor printing quality, so the thresholding problem is difficult for these images. For these images, Huang and Wang's fuzzy thresholding method has an overall better performance than other methods. For "poster 1" and "poster 2", Otsu's method and model fitting method 1 are unable to suppress background noise since the threshold values are too high. For "poster 1" the fuzzy algorithm, the minimum error method and model fitting method 2 work well and produce comparable results. For "poster 2", the minimum error method and model fitting method 2 work best for removing the background pixel, but they produce some broken characters near the right edge of the image since the thresholds are too low. For this image, the fuzzy algorithm produces some isolated noisy points in the background but does not produce broken characters. The background noise can be easily removed using a spatial filter. For the "label" image, the fuzzy method seems to work best as it gives less numbers of broken character strokes. The "face" image also contains a textured background, but all background pixel are much brighter than the foreground pixels. So all five methods work well for this image and the thresholded images have only minor differences.

3.7.2 Segmentation of Building Images

We consider the two images shown in Figs. 3.14 and 3.15. The threshold values determined from the five methods are shown in Table 3.2 and the thresholded images are shown in Figs. 3.22 and 3.23. For the "tower" image, Otsu's method, model fitting method 1 and Huang and Wang's fuzzy method produce roughly the same result. For this image, threshold values given by the minimum error method and model fitting method 2 are too low, so the right half of the tower in the thresholded images is not well extracted from the background. For the "temple" image, Otsu's method, model fitting method 1 and Huang and Wang's fuzzy method also produce similar results.

Table 3.2 Optimal threshold values determined by five thresholding methods for two building images.

Image	Thresholding Method				
	Otsu	Model Fitting 1	KI Min Error	Model Fitting 2	HW Fuzzy
Tower	127	130	93	98	129
Temple	108	107	60	59	93

Table 3.3 Optimal threshold values determined by five thresholding methods for two actor face images.

Image	Thresholding Method				
	Otsu	Model Fitting 1	KI Min Error	Model Fitting 2	HW Fuzzy
Actor 1	113	96	61	53	67
Actor 2	111	105	65	60	65

For this image, the fuzzy method performs slightly better for the roof area. Again, threshold values given by the minimum error method and model fitting method 2 are too low.

3.7.3 Segmentation of Human Images

Our experiments were performed on the "actor 1" and "actor 2" images shown in Figs. 3.16 and 3.17. The threshold values determined from the five methods are shown in Table 3.3 and the thresholded images are shown in Figs. 3.24 to 3.25. For the two images, the minimum error method, model fitting threshold method 2, and Huang and Wang's fuzzy method produce similar results. Otsu's method and model fitting method 1 produce higher threshold values and are able to extract more facial features than other methods.

We must point out here that for the two human images the purpose of thresholding is not as clear as for the document and building images. The human images contain more smooth areas and fine facial features, so their foreground and background are not well defined. One usually needs to use edge detectors and other techniques to extract facial features.

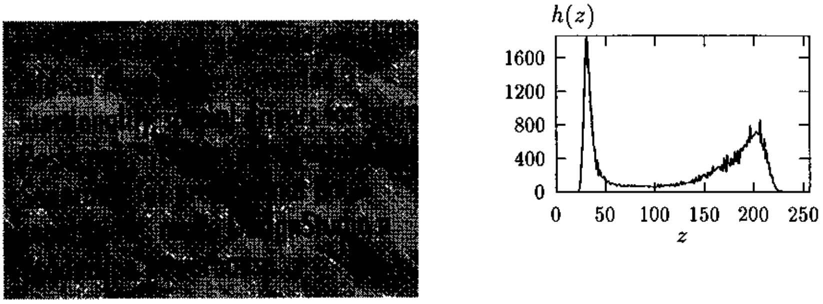


Figure 3.10 Test image "poster 1" and its gray level histogram.

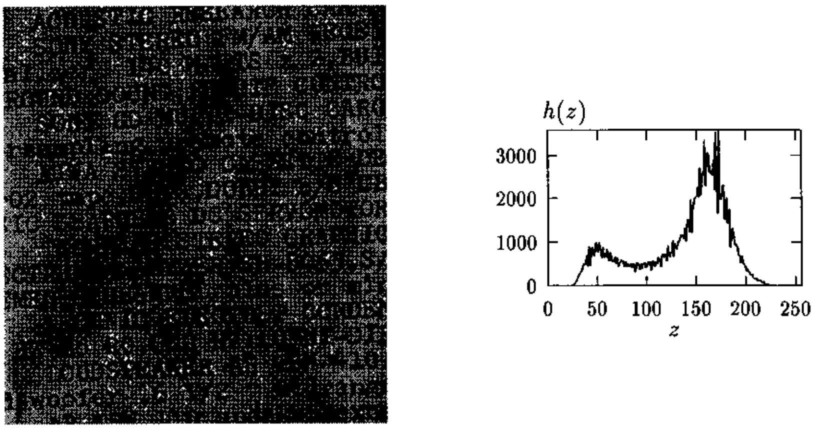


Figure 3.11 Test image "poster 2" and its gray level histogram.

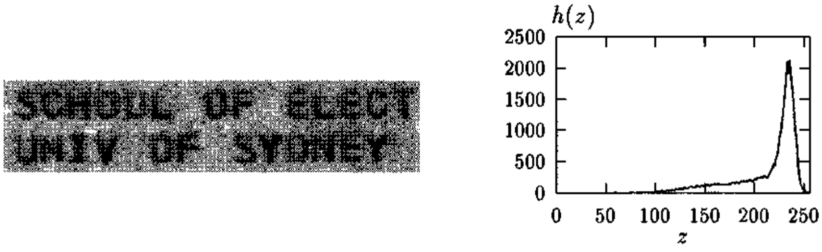


Figure 3.12 Test image "label" and its gray level histogram.

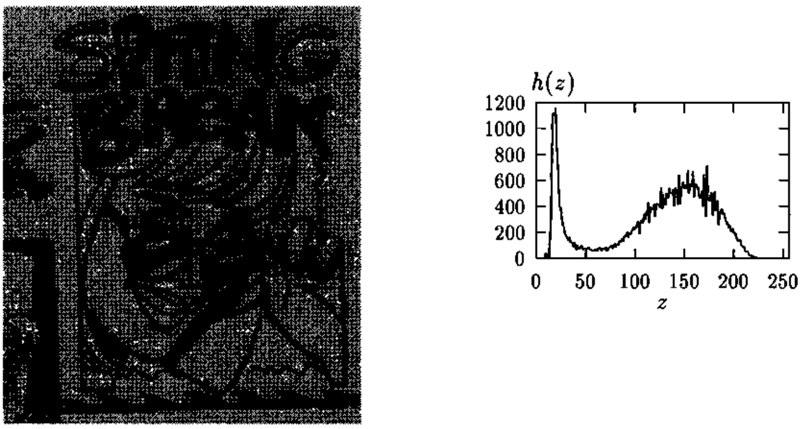


Figure 3.13 Test image "face" and its gray level histogram.

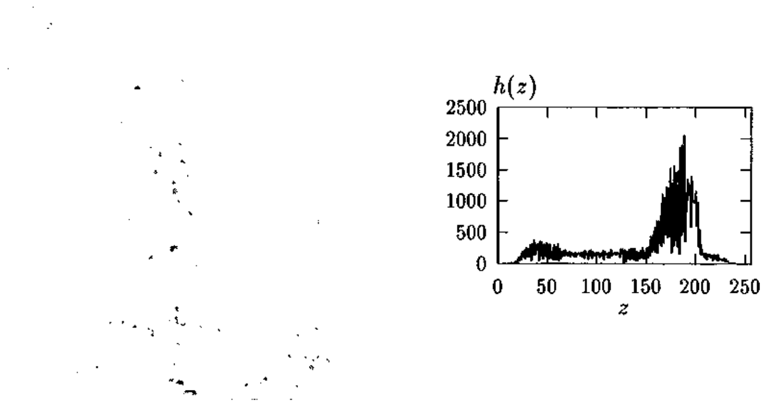


Figure 3.14 Test image "tower" and its gray level histogram.

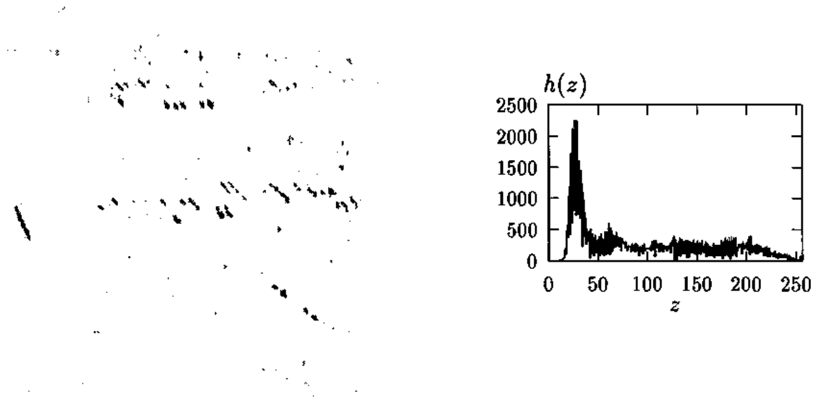


Figure 3.15 Test image "temple" and its gray level histogram.

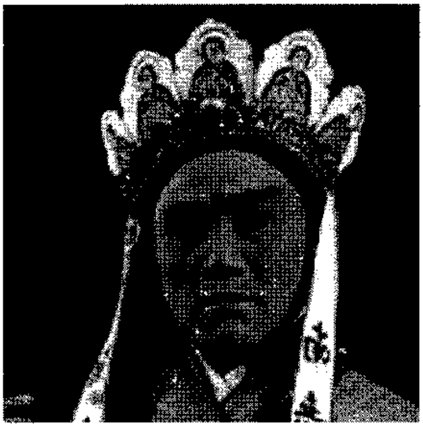
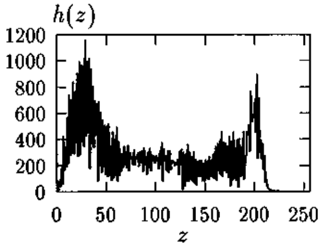



Figure 3.16 Test image "actor 1" and its gray level histogram.

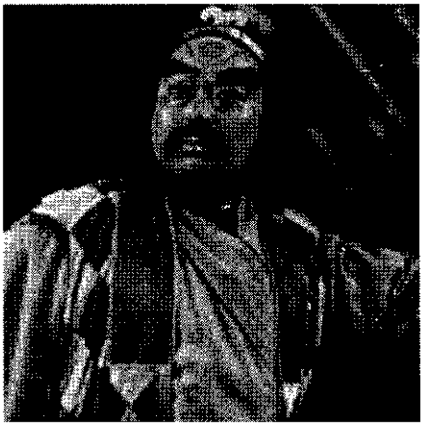
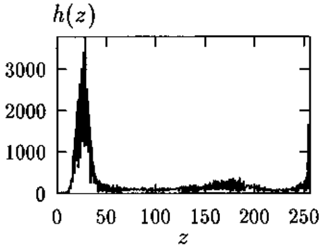
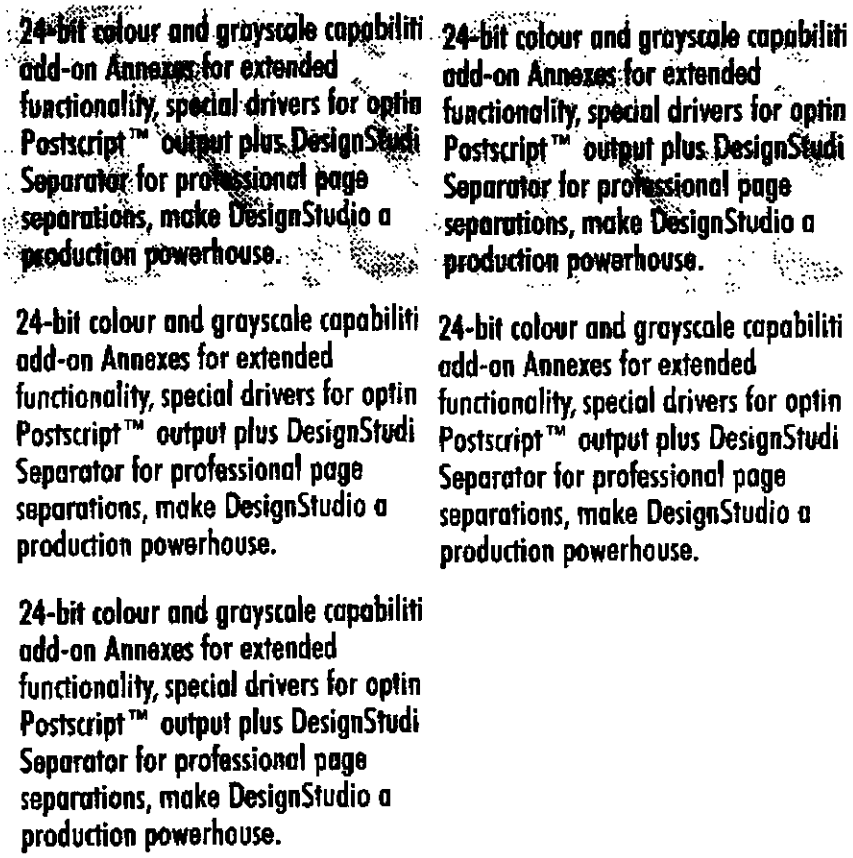



Figure 3.17 Test image "actor 2" and its gray level histogram.



24-bit colour and grayscale capabiliti
add-on Annexes for extended
functionality, special drivers for optin
Postscript™ output plus DesignStudi
Separator for professional page
separations, make DesignStudio a
production powerhouse.

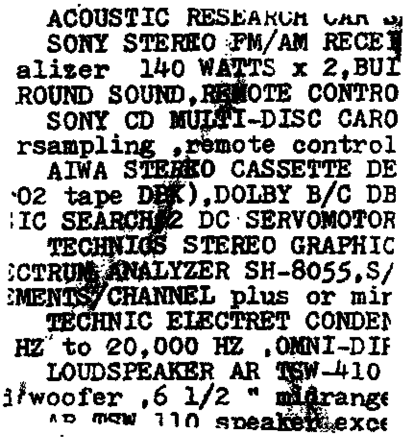
24-bit colour and grayscale capabiliti
add-on Annexes for extended
functionality, special drivers for optin
Postscript™ output plus DesignStudi
Separator for professional page
separations, make DesignStudio a
production powerhouse.

24-bit colour and grayscale capabiliti
add-on Annexes for extended
functionality, special drivers for optin
Postscript™ output plus DesignStudi
Separator for professional page
separations, make DesignStudio a
production powerhouse.

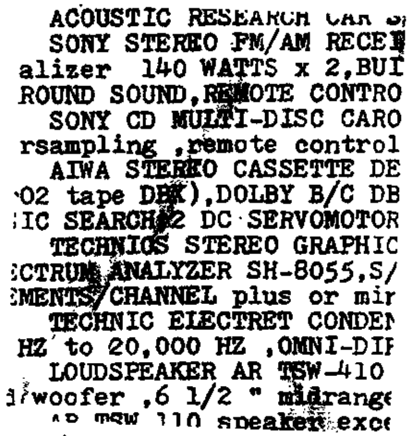
24-bit colour and grayscale capabiliti
add-on Annexes for extended
functionality, special drivers for optin
Postscript™ output plus DesignStudi
Separator for professional page
separations, make DesignStudio a
production powerhouse.

24-bit colour and grayscale capabiliti
add-on Annexes for extended
functionality, special drivers for optin
Postscript™ output plus DesignStudi
Separator for professional page
separations, make DesignStudio a
production powerhouse.

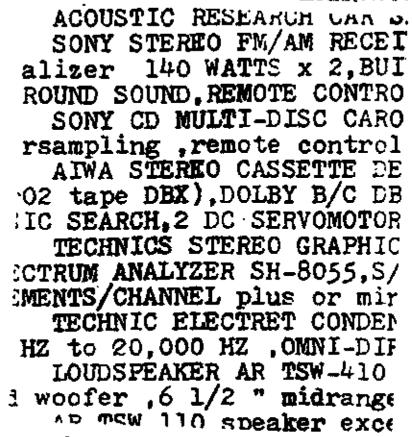
Figure 3.18 Thresholding results of "poster 1" image using Otsu's method (top left), model fitting method 1 (top right), minimum error method (middle left), model fitting method 2 (middle right), and Huang and Wang's fuzzy method (bottom left).



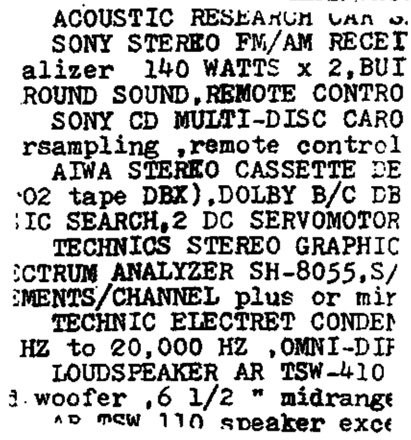
ACOUSTIC RESEARCH CAR
SONY STEREO FM/AM RECEI
alizer 140 WATTS x 2,BUI
ROUND SOUND,REMOTE CONTRO
SONY CD MULTI-DISC CARO
rsampling ,remote control
AIWA STEREO CASSETTE DE
·02 tape DBX),DOLBY B/C DB
IC SEARCH,2 DC SERVOMOTOR
TECHNICS STEREO GRAPHIC
CTRUM ANALYZER SH-8055,S/
MENTS/CHANNEL plus or mir
TECHNIC ELECTRET CONDEN
HZ to 20,000 HZ ,OMNI-DIF
LOUDSPEAKER AR TSW-410
3 woofer ,6 1/2 " midrange
AR TSW 110 speaker exce



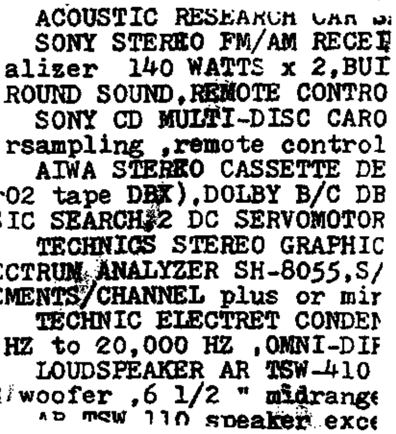
ACOUSTIC RESEARCH CAR
SONY STEREO FM/AM RECEI
alizer 140 WATTS x 2,BUI
ROUND SOUND,REMOTE CONTRO
SONY CD MULTI-DISC CARO
rsampling ,remote control
AIWA STEREO CASSETTE DE
·02 tape DBX),DOLBY B/C DB
IC SEARCH,2 DC SERVOMOTOR
TECHNICS STEREO GRAPHIC
CTRUM ANALYZER SH-8055,S/
MENTS/CHANNEL plus or mir
TECHNIC ELECTRET CONDEN
HZ to 20,000 HZ ,OMNI-DIF
LOUDSPEAKER AR TSW-410
3 woofer ,6 1/2 " midrange
AR TSW 110 speaker exce



ACOUSTIC RESEARCH CAR
SONY STEREO FM/AM RECEI
alizer 140 WATTS x 2,BUI
ROUND SOUND,REMOTE CONTRO
SONY CD MULTI-DISC CARO
rsampling ,remote control
AIWA STEREO CASSETTE DE
·02 tape DBX),DOLBY B/C DB
IC SEARCH,2 DC SERVOMOTOR
TECHNICS STEREO GRAPHIC
CTRUM ANALYZER SH-8055,S/
MENTS/CHANNEL plus or mir
TECHNIC ELECTRET CONDEN
HZ to 20,000 HZ ,OMNI-DIF
LOUDSPEAKER AR TSW-410
3 woofer ,6 1/2 " midrange
AR TSW 110 speaker exce



ACOUSTIC RESEARCH CAR
SONY STEREO FM/AM RECEI
alizer 140 WATTS x 2,BUI
ROUND SOUND,REMOTE CONTRO
SONY CD MULTI-DISC CARO
rsampling ,remote control
AIWA STEREO CASSETTE DE
·02 tape DBX),DOLBY B/C DB
IC SEARCH,2 DC SERVOMOTOR
TECHNICS STEREO GRAPHIC
CTRUM ANALYZER SH-8055,S/
MENTS/CHANNEL plus or mir
TECHNIC ELECTRET CONDEN
HZ to 20,000 HZ ,OMNI-DIF
LOUDSPEAKER AR TSW-410
3 woofer ,6 1/2 " midrange
AR TSW 110 speaker exce



ACOUSTIC RESEARCH CAR
SONY STEREO FM/AM RECEI
alizer 140 WATTS x 2,BUI
ROUND SOUND,REMOTE CONTRO
SONY CD MULTI-DISC CARO
rsampling ,remote control
AIWA STEREO CASSETTE DE
·02 tape DBX),DOLBY B/C DB
IC SEARCH,2 DC SERVOMOTOR
TECHNICS STEREO GRAPHIC
CTRUM ANALYZER SH-8055,S/
MENTS/CHANNEL plus or mir
TECHNIC ELECTRET CONDEN
HZ to 20,000 HZ ,OMNI-DIF
LOUDSPEAKER AR TSW-410
3 woofer ,6 1/2 " midrange
AR TSW 110 speaker exce

Figure 3.19 Thresholding results of "poster 2" image using Otsu's method (top left), model fitting method 1 (top right), minimum error method (middle left), model fitting method 2 (middle right), and Huang and Wang's fuzzy method (bottom left).

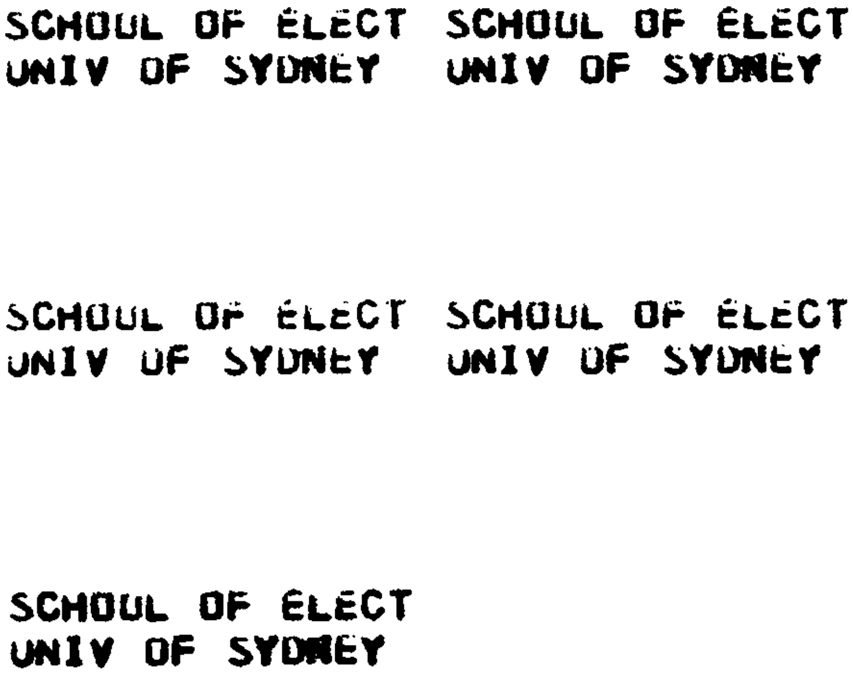


Figure 3.20 Thresholding results of "label" image using Otsu's method (top left), model fitting method 1 (top right), minimum error method (middle left), model fitting method 2 (middle right), and Huang and Wang's fuzzy method (bottom left).

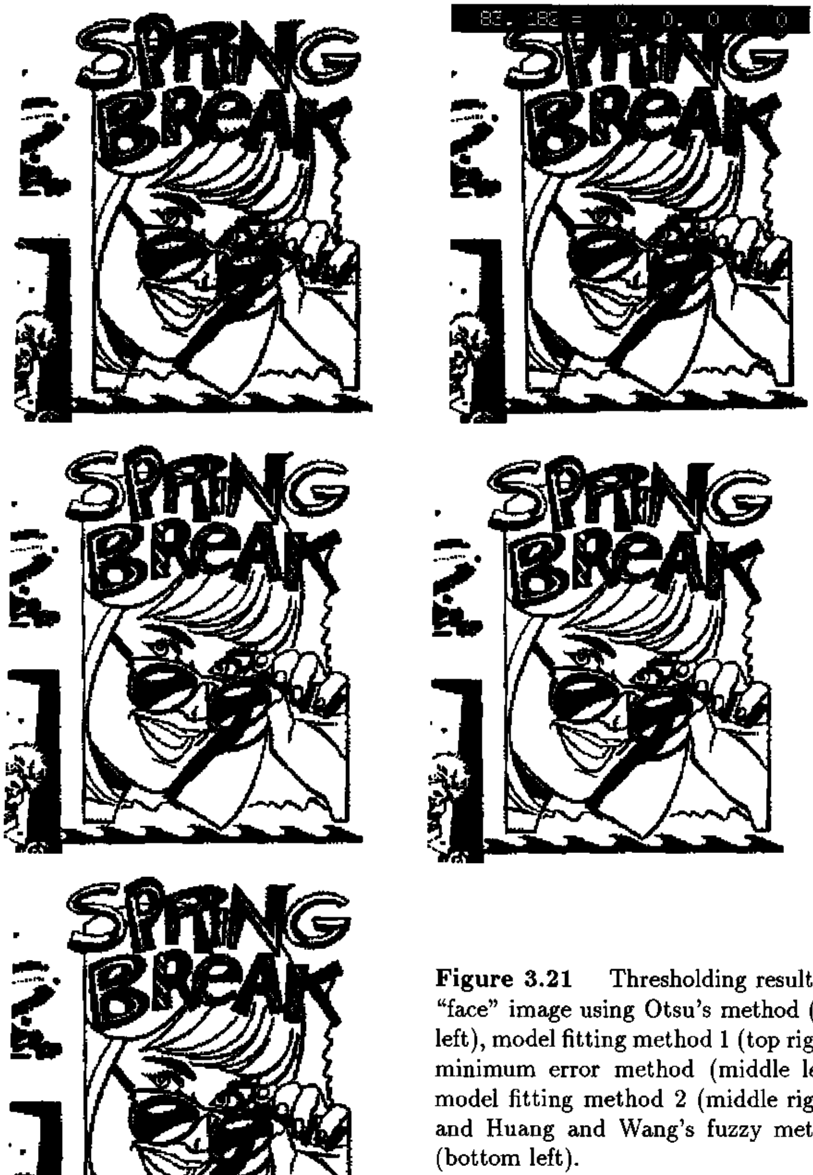


Figure 3.21 Thresholding results of "face" image using Otsu's method (top left), model fitting method 1 (top right), minimum error method (middle left), model fitting method 2 (middle right), and Huang and Wang's fuzzy method (bottom left).

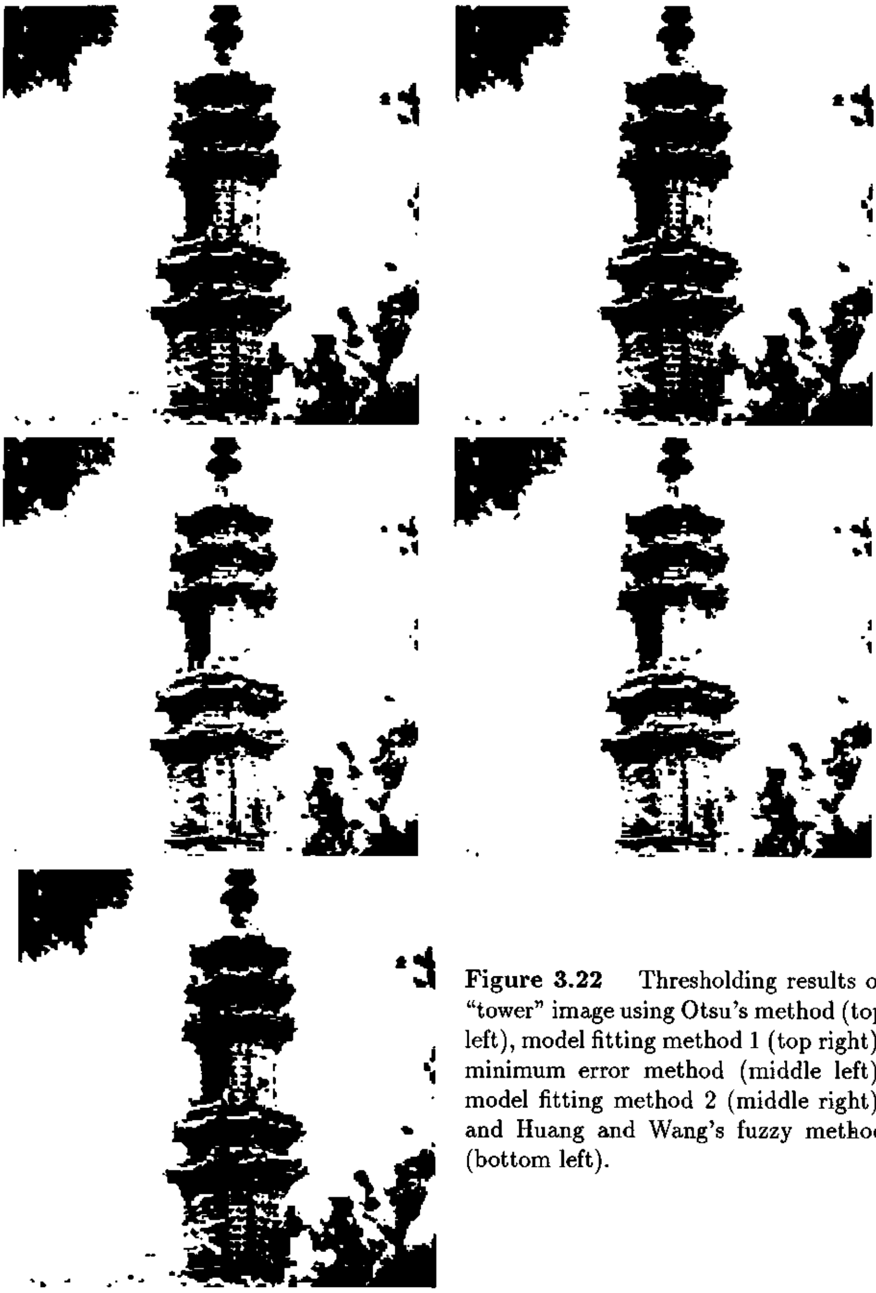


Figure 3.22 Thresholding results of "tower" image using Otsu's method (top left), model fitting method 1 (top right), minimum error method (middle left), model fitting method 2 (middle right), and Huang and Wang's fuzzy method (bottom left).

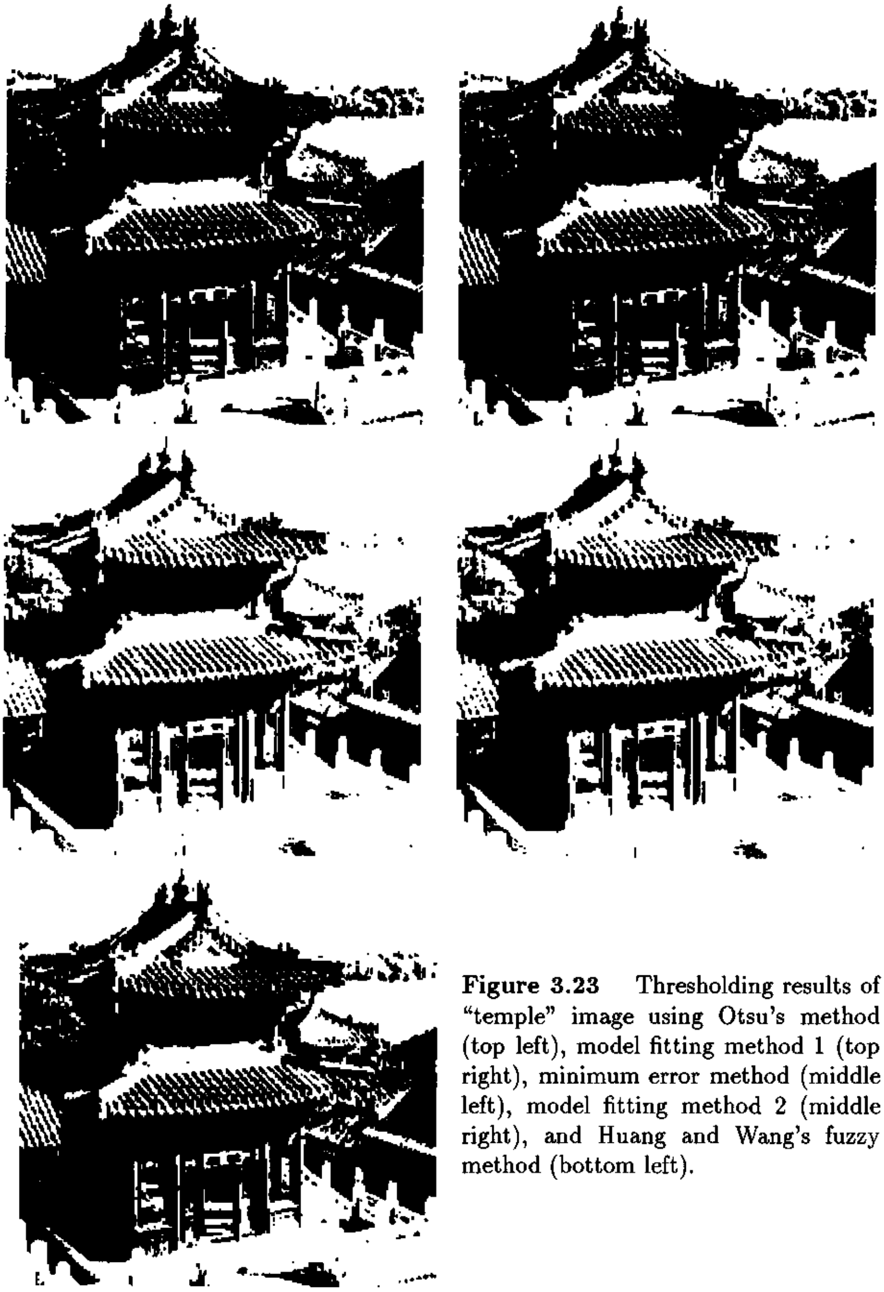


Figure 3.23 Thresholding results of "temple" image using Otsu's method (top left), model fitting method 1 (top right), minimum error method (middle left), model fitting method 2 (middle right), and Huang and Wang's fuzzy method (bottom left).

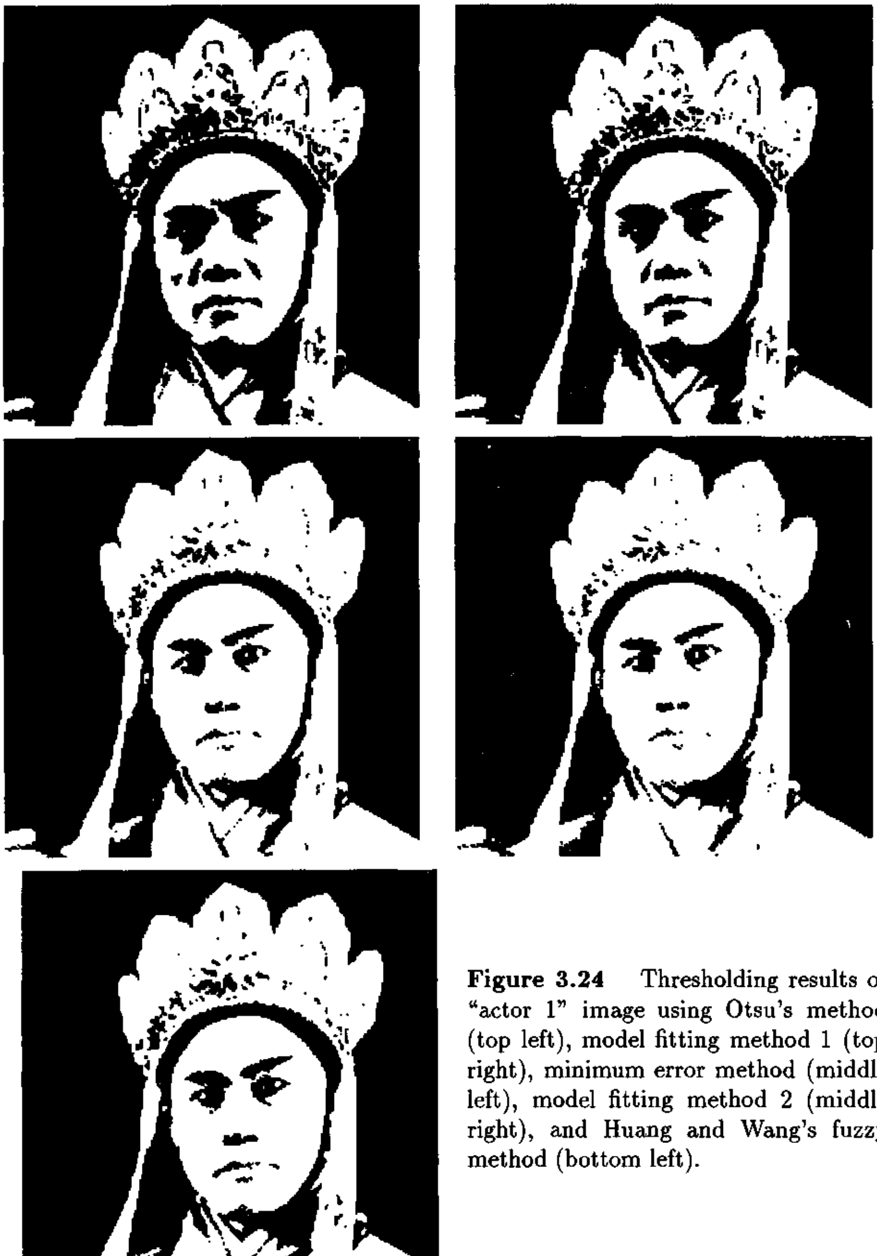


Figure 3.24 Thresholding results of "actor 1" image using Otsu's method (top left), model fitting method 1 (top right), minimum error method (middle left), model fitting method 2 (middle right), and Huang and Wang's fuzzy method (bottom left).

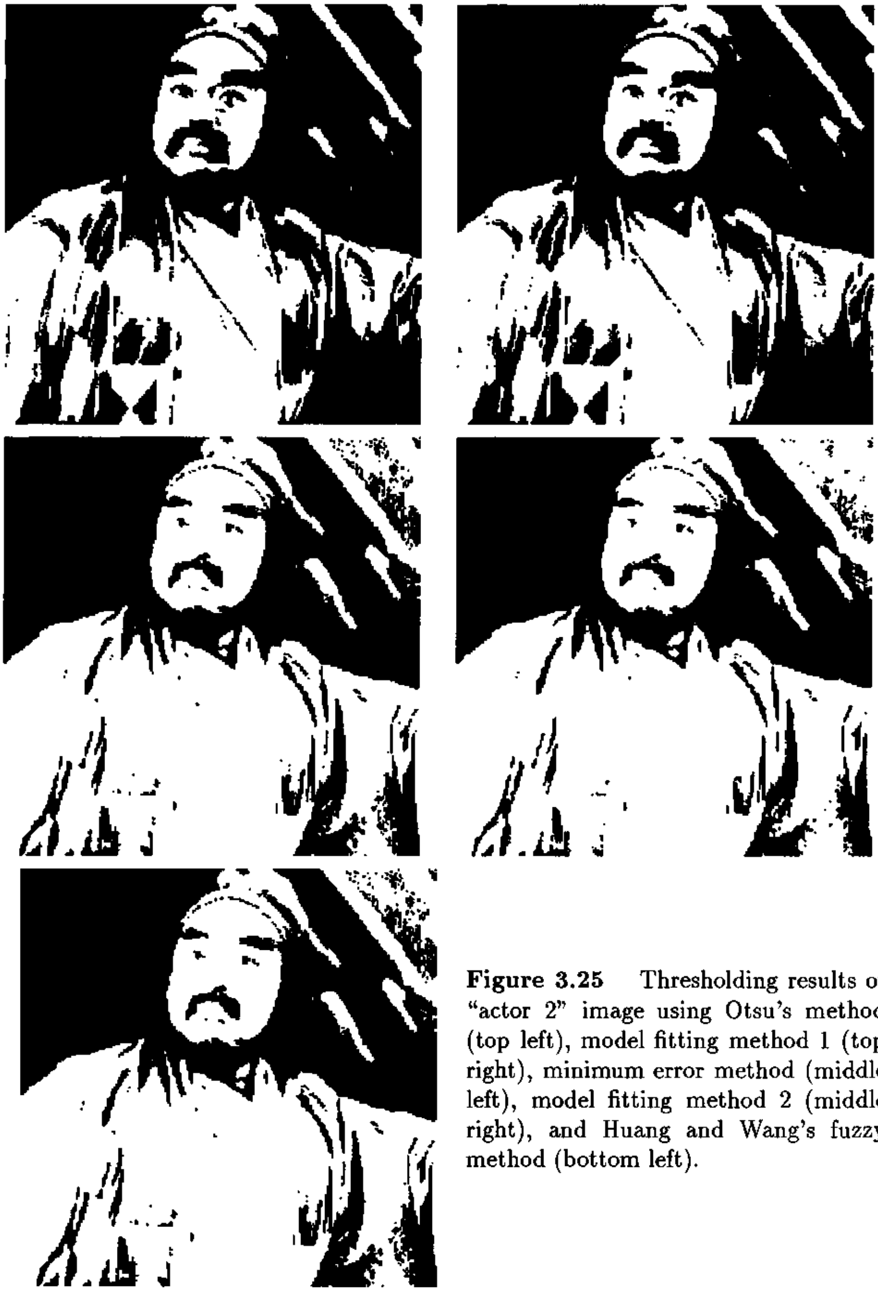


Figure 3.25 Thresholding results of "actor 2" image using Otsu's method (top left), model fitting method 1 (top right), minimum error method (middle left), model fitting method 2 (middle right), and Huang and Wang's fuzzy method (bottom left).

3.8 Concluding Remarks

Based on the experimental results described in Sections 3.7.1 to 3.7.3, we can make the following comparison among the thresholding methods:

(1) The model fitting algorithm is strongly dependent of the initial parameters because of the nonlinear nature of the procedure. The segmentation results are usually close to those obtained with the initial parameters. More complicated algorithms, such as simulated annealing and genetic algorithms, can be used for the optimization procedure to avoid the local optimal solutions, but substantially more computing time may be required.

(2) All three non-iterative methods work well if the background and foreground of an image have well separated gray level ranges.

(3) For images with textured background and poor printing qualities we have tested, Huang and Wang's fuzzy algorithm has an overall consistently better performance than the other two non-iterative algorithms.

Image thresholding is a special and the simplest case of image segmentation. It is a classification procedure which makes use of only one feature, the image pixel intensity. This method cannot be used directly for color images since the pixel value histogram is multi-dimensional. In general, more image features and more sophisticated supervised and unsupervised learning methods [42], [43], [44], such as the techniques discussed in Chapters 4 and 6, should be used to solve a segmentation problem.

Chapter 4

Fuzzy Clustering

4.1 Introduction

In general there are two classes of pattern recognition techniques: supervised methods and unsupervised methods. In a supervised method, we are given a set of training samples in different classes:

Samples in class 1: $\mathbf{x}_1^{(1)}, \mathbf{x}_2^{(1)}, \cdots, \mathbf{x}_{n_1}^{(1)}$
Samples in class 2: $\mathbf{x}_1^{(2)}, \mathbf{x}_2^{(2)}, \cdots, \mathbf{x}_{n_2}^{(2)}$
......
Samples in class c: $\mathbf{x}_1^{(c)}, \mathbf{x}_2^{(c)}, \cdots, \mathbf{x}_{n_c}^{(c)}$

where $\mathbf{x}_k^{(i)}$ represents sample k in class i. For these training data, we need to find a mapping function $\Phi(\mathbf{x}_k^{(i)})$, or to build a classifier, which can be a set of fuzzy rules, a neural network, a decision tree, or simply a set of mathematical equations, so that $\Phi(\mathbf{x}_k^{(i)}) = i$. Once the mapping function is determined, it can be used to classify an unseen sample $\mathbf{x}$. The class label of $\mathbf{x}$ is simply $\Phi(\mathbf{x})$.

In an unsupervised method, we do not have training samples, and in some cases, we even do not know the exact number of classes. In this case, we are given a set of unlabeled data:

$$\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n$$

Our task is to divide these data into several groups according to a similarity measure or inherent structure of the data. Such grouping can be done using a clustering procedure. In a hard clustering procedure, a sample is either assigned to or not assigned to a group, so a clear partition is made. In a fuzzy clustering procedure, a sample is assigned a membership function for each of the groups, so a fuzzy partition is made. The membership values play an important role in the clustering process and they make the classification procedure more flexible and robust to deal with noisy and uncertain data.

In this chapter, we consider unsupervised pattern recognition techniques. In Sections 4.2 and 4.3, we describe the hard and fuzzy c-means algorithms, which can be used for clustering when the number of clusters is known. In Section 4.4, we compare the hard and fuzzy clustering algorithms and show that the fuzzy algorithm is more flexible for solving a document segmentation problem. In Section 4.5, we describe three cluster validity measures which can be used to analyze the quality of fuzzy clustering procedures. In Section 4.6, we apply the fuzzy clustering algorithms to several real world image processing and pattern recognition problems.

4.2 C-means Algorithm

We have briefly discussed the c-means clustering algorithm in Section 2.3.1 where the algorithm was used to cluster data to obtain the cluster centers and corresponding variances for generating membership functions. In this section, we bring up this algorithm as an iterative optimization procedure to be used for classification. We will discuss the algorithm in the manner that it can be easily compared with the fuzzy c-means algorithm which will be discussed in the next section.

Let

$$X = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \tag{4.1}$$

be a set of samples to be divided (clustered) into c classes. A clustering process can be considered as an iterative optimization procedure. Suppose that we already have partitioned the samples into c classes, which can be done initially by random assignment. Our task is to adjust the partition so that some kind of similarity measure is optimized. In the (hard) c-means algorithms, the similarity measure is calculated in terms of the Euclidean distance between each sample and the center of the class to which the sample belongs. The criterion function used for the clustering process is [34]

$$J(V) = \sum_{k=1}^{n} \sum_{\mathbf{x}_k \in C_i} |\mathbf{x}_k - \mathbf{v}_i|^2 \tag{4.2}$$

where $\mathbf{v}_i$ is the sample mean or the center of samples of cluster i, and $V = \{\mathbf{v}_1, \cdots, \mathbf{v}_c\}$.

To improve the similarity of the samples in each cluster, we can minimize this criterion function so that all samples are more compactly distributed around their cluster centers. Setting the derivative of $J(V)$ with respect to $\mathbf{v}_i$ to zero, we obtain

$$\frac{\partial J(V)}{\partial \mathbf{v}_i} = \sum_{k=1}^{n} \sum_{\mathbf{x}_k \in C_i} (\mathbf{x}_k - \mathbf{v}_i) = 0$$

Thus, the optimal location of cluster center $\mathbf{v}_i$ is

$$\mathbf{v}_i = \frac{1}{n_i} \sum_{\mathbf{x}_k \in C_i} \mathbf{x}_k \tag{4.3}$$

where n_i is the number of samples in class i and C_i contains all samples in class i.

Now, based on the information of initial clusters and their center positions, we can regroup the samples so that the criterion function is minimized. In Eq. (4.2), each sample $\mathbf{x}_k$ appears only once, that is, it is associated with only one cluster center. Obviously, the criterion function is minimized if each sample is associated with its closest cluster center. So we reassign $\mathbf{x}_k$ to class i so that $(\mathbf{x}_k - \mathbf{v}_j)^2$ is minimum when $j = i$.

Once the samples are regrouped, the cluster centers need to be recomputed to minimize $J(V)$. For the new cluster centers, we can again regroup the samples to reduce $J(V)$. The process can be repeated until J cannot be reduced any further.

In summary, the c-means clustering procedure consists of the following steps:

1. Determine the number of clusters c.
2. Partition the input samples into c clusters based on an approximation. If no rule of approximation exists, the samples can be partitioned randomly.
3. Compute the cluster centers.
4. Assign each input sample to the class of the closet cluster center.
5. Repeat steps 3 and 4 until no change in J can be made.

4.3 Fuzzy C-means Algorithm

In the hard clustering process, each data sample is assigned to only one cluster and all clusters are regarded as disjoint gatherings of the data set. In practice, however, there are many cases in which the clusters are not completely disjointed and data could be classified as belonging to one cluster almost as well as to another. Such a situation cannot be catered for by a crisp classification process. Therefore, the separation of the clusters becomes a fuzzy notion, and the representations of real data structures can then be more accurately handled by fuzzy clustering methods. In these cases, it is necessary to describe the data structure in terms of fuzzy clusters.

The fuzzy c-means (FCM) algorithm is the best known and the most widely used fuzzy clustering technique. This algorithm is developed based on iterative minimization of the following criterion function [2], [5], [45]

$$J(U,V) = \sum_{i=1}^{c} \sum_{k=1}^{n} u_{ik}^{m} |\mathbf{x}_k - \mathbf{v}_i|^2 \tag{4.4}$$

where

- $\mathbf{x}_1, \cdots, \mathbf{x}_n$ are n data sample vectors;
- $V = \{\mathbf{v}_1, \cdots, \mathbf{v}_c\}$ are cluster centers;

- $U = [u_{ik}]$ is a $c \times n$ matrix, where u_{ik} is the ith membership value of the kth input sample $\mathbf{x}_k$, and the membership values satisfy the following conditions

$$0 \leq u_{ik} \leq 1 \qquad i = 1, 2, \cdots, c; \;\; k = 1, 2, \cdots, n \tag{4.5}$$

$$\sum_{i=1}^{c} u_{ik} = 1 \qquad k = 1, 2, \cdots, n \tag{4.6}$$

$$0 < \sum_{k=1}^{n} u_{ik} < n \qquad i = 1, 2, \cdots, c \tag{4.7}$$

- $m \in [1, \infty)$ is an exponent weight factor.

The objective function is the sum of the squared Euclidean distances between each input sample and its corresponding cluster center, with the distances weighted by the fuzzy memberships.

The algorithm is iterative and makes use of the following equations:

$$\mathbf{v}_i = \frac{1}{\sum_{k=1}^{n} u_{ik}^m} \sum_{k=1}^{n} u_{ik}^m x_{ik} \qquad i = 1, 2, \cdots, c \tag{4.8}$$

$$u_{ik} = \frac{\left[\dfrac{1}{|\mathbf{x}_k - \mathbf{v}_i|^2}\right]^{1/(m-1)}}{\sum_{j=1}^{c} \left[\dfrac{1}{|\mathbf{x}_k - \mathbf{v}_j|^2}\right]^{1/(m-1)}} \qquad i = 1, 2, \cdots, c; \;\; k = 1, 2, \cdots, n \tag{4.9}$$

For calculation of a cluster center, all input samples are considered and the contributions of the samples are weighted by the membership values. For each sample, its membership value in each class depends on its distance to the corresponding cluster center. The weight factor m reduces the influence of small membership values. The larger the value of m, the smaller the influence of samples with small membership values.

The FCM clustering procedure consists of the following steps:

1. Initialize $U^{(0)}$ randomly or based on an approximation; initialize $V^{(0)}$ and calculate $U^{(0)}$. Set the iteration counter $\alpha = 1$. Select the number of class centers c and choose the exponent weight m.

2. Compute the cluster centers. Given $U^{(\alpha)}$, calculate $V^{(\alpha)}$ according to Eq. (4.8).

3. Update the membership values. Given $V^{(\alpha)}$, calculate $U^{(\alpha)}$ according to Eq. (4.9).

4. Stop the iteration if
$$\max \mid u_{ik}^{(\alpha)} - u_{ik}^{(\alpha-1)} \mid \leq \epsilon \tag{4.10}$$
else let $\alpha = \alpha + 1$ and go to Step 2, where ϵ is the pre-specified small number representing the smallest acceptable change in U.

4.4 Comparison between Hard and Fuzzy Clustering Algorithms

The hard c-means clustering algorithms can be considered as a special case of the fuzzy c-means clustering algorithms. In Eq. (4.4), if u_{ik} is 1 for only one class and zero for all other classes, then the criterion function $J(U, V)$ used in the FCM clustering algorithm is the same as the criterion function $J(V)$ used in the hard c-means cluster algorithm. The use of membership values provides more flexibility and makes the clustering results more useful in practical applications. We show below an example of document image segmentation to compare the two clustering algorithms.

We consider a document image shown in Fig. 4.1 and our task here is to segment the image, that is, to separate the background and foreground of the image. This image is difficult to deal with since it has a non-uniform background. Although there is a valley in the gray level histogram of the image, which indicates that there are two groups (clusters) of pixels in the image, the valley is not deep, that is, two groups overlap each other in terms of gray level values. As a result, the segmentation is very sensitive to the threshold value if a thresholding method is used to segment the image. Several image thresholding methods are described in Chapter 3. The best thresholding result of the image is shown in Fig. 4.2. This image is considered and all results obtained using different thresholding methods are shown in Chapter 3.

The thresholding method has some limitations since it can deal with only one feature of the image, the pixel gray level. To take the image geometric properties into account, we can make use of some spatial features. To do this, however, we need to solve a problem of classification in a multi-dimensional feature space and the simple threshold method can no longer be used. The clustering methods described in this section can be used to solve this problem. In our experiment, we made use of seven features at each pixel, which include the pixel gray level, mean and standard deviation of pixel gray levels in a neighborhood of 5 by 5 pixels around each pixel, and edge intensities along the horizontal, the vertical and two diagonal directions. Using the c-means algorithm, we obtained the segmentation result shown in Fig. 4.3. In general, the c-means algorithm performs better than the simple thresholding method as the segmentation result is less sensitive to background variations. However, the characters extracted using the c-means algorithm appear to be blurred. We show below that the blurring problem can be solved if we use the FCM clustering algorithm.

Now we consider the FCM clustering method to segment the image. At each pixel, we have two membership values, one representing the degree of certainty of a pixel belonging to background and the other representing the degree of certainty of a pixel belonging to foreground. Since the sum of two membership values must be equal to one, they are not independent of each other. In general, for a c class problem, only $c - 1$ membership values for each input sample are independent. For the document image, we make use of the background membership values, which are displayed as an image in Fig. 4.4 after scaling. The gray level histogram of the membership image

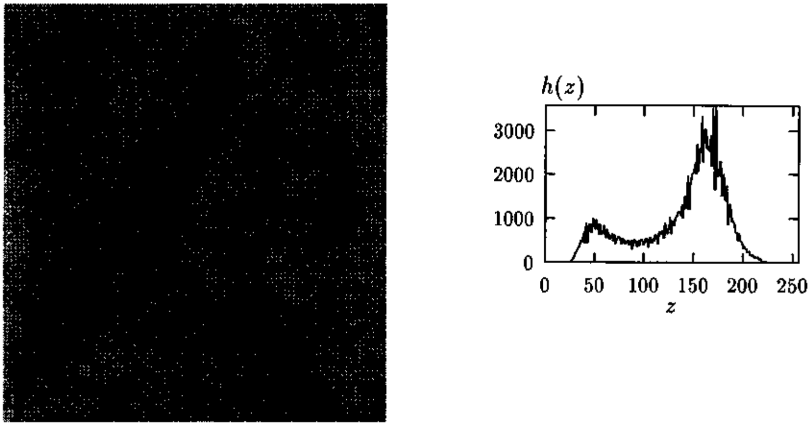
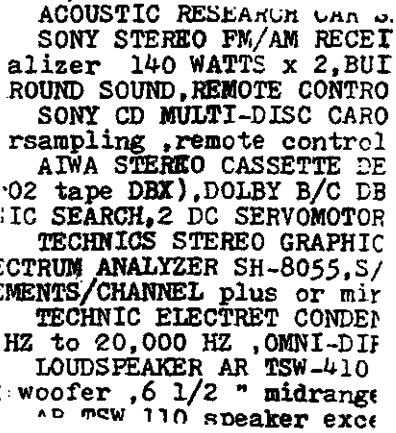
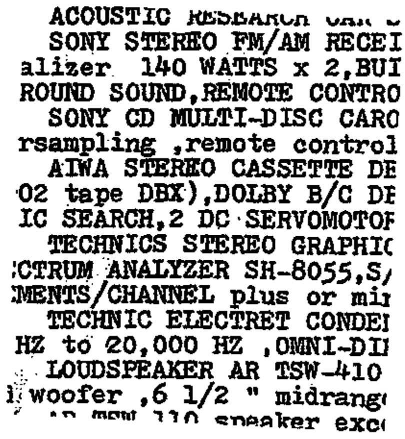


Figure 4.1 A document image and its gray level histogram.



ACOUSTIC RESEARCH [illegible]
SONY STEREO FM/AM RECEI
alizer 140 WATTS x 2,BUI
ROUND SOUND,REMOTE CONTRO
SONY CD MULTI-DISC CARO
rsampling ,remote control
AIWA STEREO CASSETTE DE
·02 tape DBX),DOLBY B/C DB
IC SEARCH,2 DC SERVOMOTOR
TECHNICS STEREO GRAPHIC
CTRUM ANALYZER SH-8055,S/
MENTS/CHANNEL plus or mir
TECHNIC ELECTRET CONDEN
HZ to 20,000 HZ ,OMNI-DIR
LOUDSPEAKER AR TSW-410
woofer ,6 1/2 " midrange
[illegible] 110 speaker exce

Figure 4.2 Segmentation result obtained using a thresholding method.



ACOUSTIC RESEARCH [illegible]
SONY STEREO FM/AM RECEI
alizer 140 WATTS x 2,BUI
ROUND SOUND,REMOTE CONTRO
SONY CD MULTI-DISC CARO
rsampling ,remote control
AIWA STEREO CASSETTE DE
02 tape DBX),DOLBY B/C DB
IC SEARCH,2 DC SERVOMOTOR
TECHNICS STEREO GRAPHIC
CTRUM ANALYZER SH-8055,S/
MENTS/CHANNEL plus or mir
TECHNIC ELECTRET CONDEN
HZ to 20,000 HZ ,OMNI-DIR
LOUDSPEAKER AR TSW-410
woofer ,6 1/2 " midrange
[illegible] 110 speaker exce

Figure 4.3 Segmentation result obtained using the c-means algorithm.

is also shown in Fig. 4.4. It is interesting to see that this histogram is similar to that in Fig. 4.1 except that its valley is much deeper and wider. Thus, we segment the image reliably now using a thresholding method. With a threshold of 91, we obtained the segmentation result shown in Fig. 4.5. Compared with the image in Fig. 4.3 obtained using the hard c-means algorithm, the segmented image in Fig. 4.5 is slightly better as it does not have isolated noisy pixels in the background. However, with the membership values as a gray scale image, we have more flexibilities. When lowering the threshold to 42, we obtained the segmented image shown in Fig. 4.6. Now this image is no longer blurred. In general, we can change the threshold value to change the width of character strokes in the segmented image. This can be important for some optical character recognition (OCR) systems as the recognition result may depend on the width of character strokes.

4.5 Cluster Validity

In practical applications, we need a cluster validity method to measure the quality of the clustering result. The quality of a clustering process depends on many factors, such as the method of initialization, the choice of the number of classes c, and the clustering method. As shown in Section 4.4, the FCM clustering algorithm is more flexible than the hard c-means algorithm, so we consider the cluster validity problem for the FCM algorithm only in this section. The method of initialization requires a good estimate of the clusters and is application dependent, so the cluster validity problem is reduced to the choice of an optimal number of classes c.

Several cluster validity measures have been developed in the past [2] [45], [46]. In this section, we describe three of these measures: partition coefficient [45], partition entropy [45] and compactness and separation validity function [47].

The partition coefficient is defined as [45]

$$F(U,c) = \frac{1}{n}\sum_{i=1}^{c}\sum_{k=1}^{n}(\mu_{ik})^2 \tag{4.11}$$

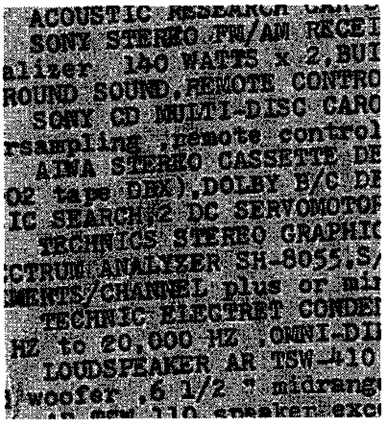
Suppose that Ω_c represents the clustering result, then the optimal choice of c is given by

$$\max_{c}\left\{\max_{\Omega_c} F(U,c)\right\} \quad c = 2, \cdots, n-1 \tag{4.12}$$

The partition coefficient measures the closeness of all input samples to their corresponding cluster centers. If each sample is closely associated with only one cluster, that is, if for each k, u_{ik} is large for only one i value, then the uncertainty of the data is small, which corresponds to a large $F(U,c)$ value.

The partition entropy is defined as [45]

$$H(U,c) = -\frac{1}{n}\sum_{i=1}^{c}\sum_{k=1}^{n}\mu_{ik}\log(\mu_{ik}) \tag{4.13}$$



ACOUSTIC
SONY STEREO FM/AM RECEI
alizer 140 WATTS x 2,BUI
ROUND SOUND,REMOTE CONTRO
SONY CD MULTI-DISC CARO
rsampling ,remote control
AIWA STEREO CASSETTE DE
02 tape DBX),DOLBY B/C DI
IC SEARCH,2 DC SERVOMOTOF
TECHNICS STEREO GRAPHIC
CTRUM ANALYZER SH-8055,S/
MENTS/CHANNEL plus or mi
TECHNIC ELECTRET CONDEI
HZ to 20,000 HZ ,OMNI-DI
LOUDSPEAKER AR TSW-410
woofer ,6 1/2 " midrang
speaker exc

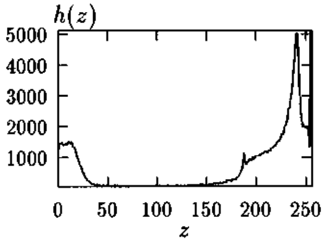
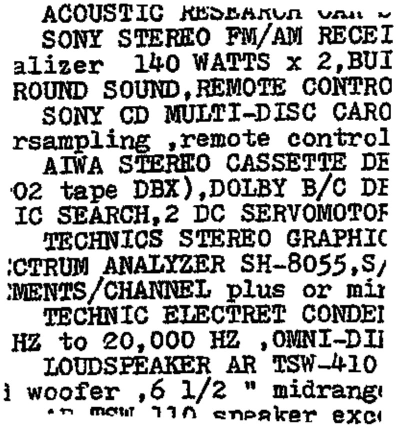
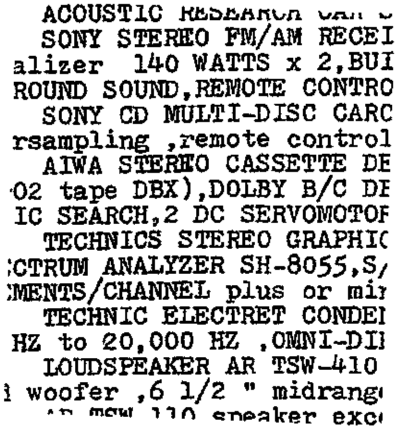


Figure 4.4 Fuzzy membership values displayed as an image and the gray level histogram of the image.



ACOUSTIC
SONY STEREO FM/AM RECEI
alizer 140 WATTS x 2,BUI
ROUND SOUND,REMOTE CONTRO
SONY CD MULTI-DISC CARO
rsampling ,remote control
AIWA STEREO CASSETTE DE
02 tape DBX),DOLBY B/C DI
IC SEARCH,2 DC SERVOMOTOF
TECHNICS STEREO GRAPHIC
CTRUM ANALYZER SH-8055,S/
MENTS/CHANNEL plus or mi
TECHNIC ELECTRET CONDEI
HZ to 20,000 HZ ,OMNI-DI
LOUDSPEAKER AR TSW-410
woofer ,6 1/2 " midrang
speaker exc

Figure 4.5 Segmentation result obtained from Fig. 4.4 with a threshold of 91.



ACOUSTIC
SONY STEREO FM/AM RECEI
alizer 140 WATTS x 2,BUI
ROUND SOUND,REMOTE CONTRO
SONY CD MULTI-DISC CARC
rsampling ,remote control
AIWA STEREO CASSETTE DE
02 tape DBX),DOLBY B/C DI
IC SEARCH,2 DC SERVOMOTOF
TECHNICS STEREO GRAPHIC
CTRUM ANALYZER SH-8055,S/
MENTS/CHANNEL plus or mi
TECHNIC ELECTRET CONDEI
HZ to 20,000 HZ ,OMNI-DI
LOUDSPEAKER AR TSW-410
woofer ,6 1/2 " midrang
speaker exc

Figure 4.6 Segmentation result obtained from Fig. 4.4 with a threshold of 42.

The optimal choice of c is given by

$$\min_{c}\left\{\min_{\Omega_c} H(U,c)\right\} \quad c = 2,\cdots,n-1 \tag{4.14}$$

When all μ_{ik}'s have values close to 0.5, which represents a high degree of fuzziness of the clusters, $H(U,c)$ is large and thus indicates a poor clustering result. On the other hand, if all μ_{ik}'s have values close to 0 or 1, $H(U,c)$ is small and indicates a good clustering result.

The compactness and separation validity function is defined as:

$$S(U,c) = \frac{\frac{1}{n}\sum_{i=1}^{c}\sum_{k=1}^{n}\mu_{ik}^2 \mid \mathbf{x}_k - \mathbf{v}_i \mid^2}{\min\limits_{i,j} \mid \mathbf{v}_i - \mathbf{v}_j \mid^2} \tag{4.15}$$

The optimal choice of c is given by

$$\min_{c}\left\{\min_{\Omega_c} S(U,c)\right\} \quad c = 2,\cdots,n-1 \tag{4.16}$$

$S(U,c)$ is the ratio between the average distance of input samples to their corresponding cluster centers and the minimum distance between cluster centers. A good cluster procedure should make all input samples as close to their cluster centers as possible and all cluster centers separated as far as possible.

As shown in [47] and in an experiment presented in the next section, the compactness and separation validity function seems to work better for image segmentation problems.

4.6 Applications

4.6.1 Characterization of Tissues in MR Images

Characterization of brain tissues in magnetic resonance (MR) images is useful for some medical diagnosis procedures. The task here is to classify image pixels into several meaningful categories. This can be carried out using the FCM clustering technique.

We consider two MR images shown on the left in Figs. 4.7 and 4.8. For each image pixel, we extract seven features, which include pixel intensity, local mean and variance computed in a neighborhood of 5 by 5 pixels, and edge values given by the Sobel operators along four different directions. For each image, we chose the number of classes $c = 4$, corresponding to the background of the image, and three types of brain tissues, white matter, gray matter and cerebrospinal fluid (CSF). The segmented images are shown on the right in Figs. 4.7 and 4.8 respectively. In Fig. 4.7, three kinds of brain tissues are well separated. In Fig. 4.8, the tumor area has been isolated from the surrounding tissues in the image.

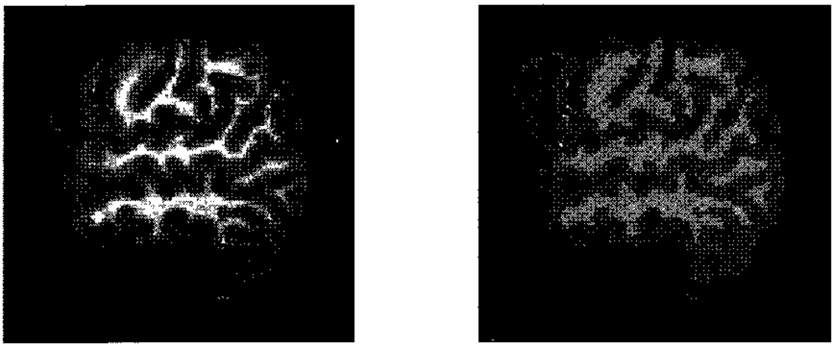


Figure 4.7 An MR image of brain tissues (left) and its segmentation result (right).

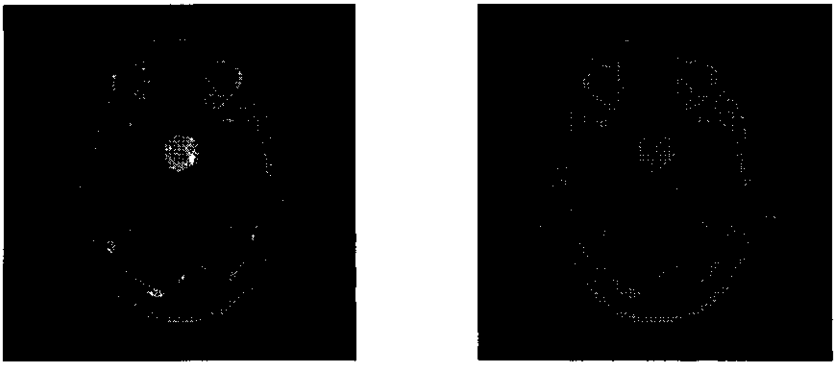


Figure 4.8 A brain MR image (left) and (left) and its segmentation result (right). The tumor area in the image has been isolated from the surrounding tissues.

4.6.2 Segmentation of Natural Color Images

We consider a color image which contains three types of fruit, banana, orange and kiwi. The intensity and the r, g, b components of the image are shown in Fig. 4.9. The intensity image is defined as the average of three component images. The three pieces of fruit and the background (at the bottom of the image between the kiwi and orange) have four different colors, yellow, orange, green and purple respectively. On the skin of the orange, and at the center and along the radial directions of both the orange and the kiwi, there are some white areas. Altogether, there are roughly five classes of image pixels, white, yellow, orange, green and purple.

Our aim here is to extract the five classes in the image. We apply the FCM clustering algorithm to the image to carry out this task. We make use of three features corresponding to r, g, b, values of the image pixels. For $c = 5$, the segmentation result is shown in Fig. 4.10. We can see that all five classes have been well extracted from the image.

In this image segmentation problem, we know that the optimal number of classes $c = 5$. Now we would like to test the performance of the cluster validity measures described in Section 4.5. In our experiment we changed c from 2 to 8 and obtained the values of cluster validity measures shown in Table 4.1.

Table 4.1 Fuzzy measures for the color image clustering problem.

c	F(U, c)	H(U, c)	S (u, c)
2	0.724391	0.426785	0.218259
3	0.680535	0.569435	0.209790
4	0.604033	0.748381	0.255265
5	0.625072	0.768383	0.137158
6	0.612068	0.826982	0.109354
7	0.573524	0.919343	0.299247
8	0.553558	0.989649	0.220133

The compactness and separation validity function works better than the other two methods for this image. It produces the optimal number of classes $c = 6$, which is close to $c = 5$ as we have explained before. In fact, in the segmented image for $c = 6$ in Fig. 4.11, the shadow of the kiwi on the orange is classified as another class, which is a quite reasonable result.

In the validity measures described in Section 4.5, only the membership values of the input samples are used. In practical applications, we can use additional criteria to check the optimal number of classes obtained from the validity measures. For example, in image segmentation, we can check the uniformity of the segmented areas to take into account the spatial information in addition to the membership values.

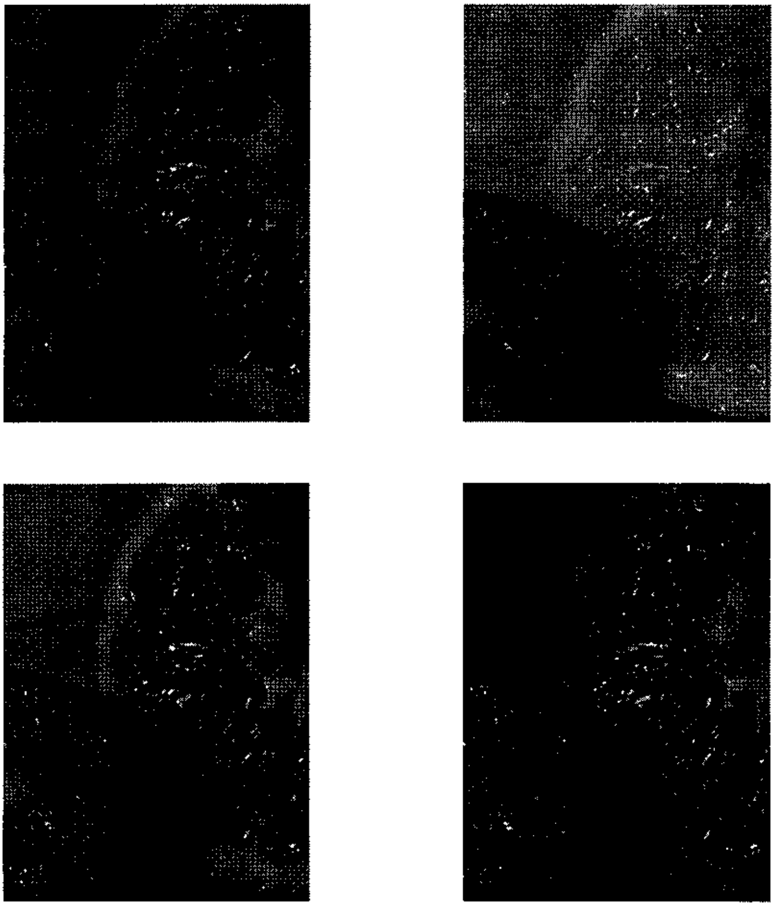


Figure 4.9 The intensity (top left) and r (top right), g (bottom left), b (bottom right) components of a color image.

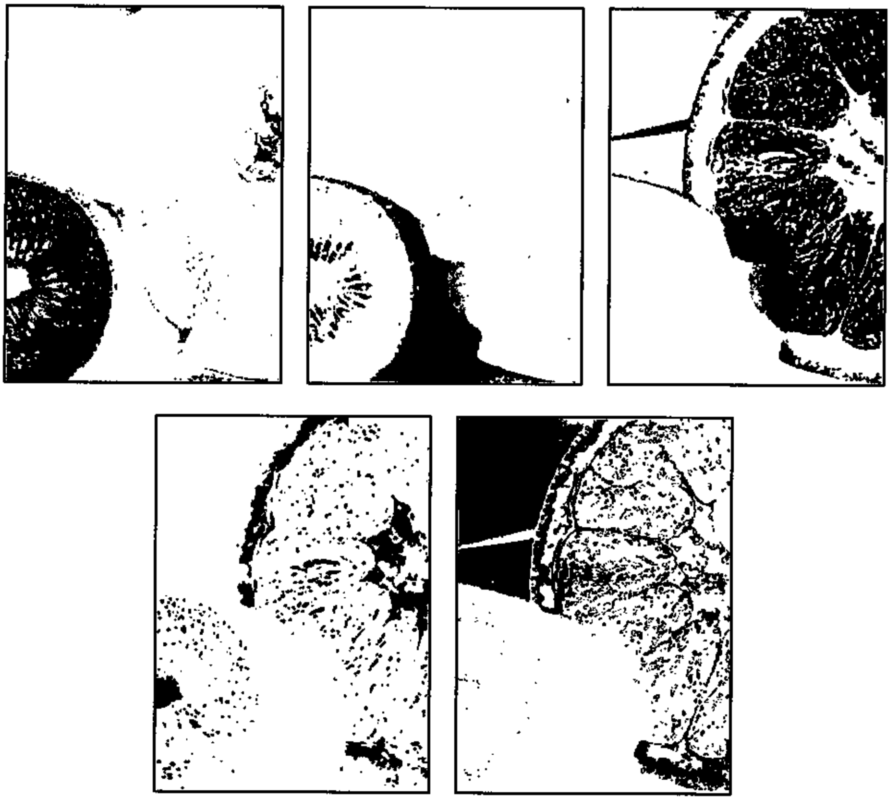
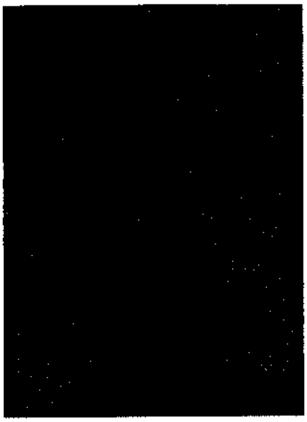
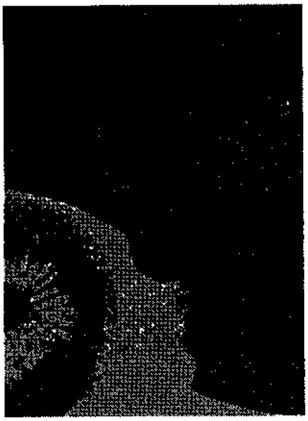


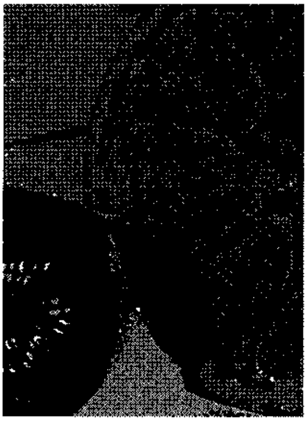
Figure 4.10 Five classes of pixels obtained using the FCM clustering algorithm with $c = 5$. These five clusters correspond to the green area of the kiwi, the purple area of background, the orange area of the orange, the white area in the kiwi and the orange and the yellow area of the banana.

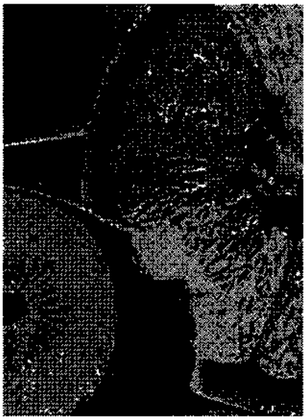

(a)

(b)


(c)

(d)


(e)


(f)

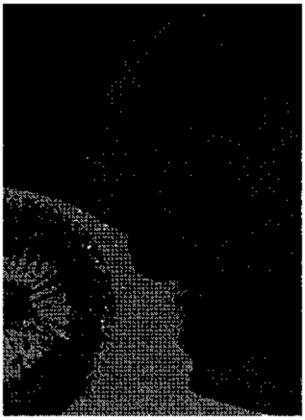

(g)

Figure 4.11 Results of segmentation of the color image in Fig. 4.9 using the FCM clustering algorithms with (a) c = 2, (b) c = 3, (c) c = 4, (d) c = 5, (e) c = 6, (f) c = 7, and (g) c = 8. The classes in each image are shown with different gray levels.

4.6.3 Fuzzy Parameter Identification

Estimation of resources is essential for the purpose of economic planning in the mining industry. In practice, the true value of an ore-body is never known until the mining process has been carried out. This uncertainty has appealed for a need to develop resource-estimation methodologies by which financiers and managers can be assisted to evaluate their mining project with a minimum risk of incorrect prediction. There are various methods developed for this purpose, such as geometrical methods, distance-weighting methods, and geostatistical methods. Each of these has its own advantages and disadvantages [48], [49], [50]. An attempt is made in this chapter to introduce an approach for ore-grade estimation based on an unconventional concept of uncertainty known as fuzziness. The concept of imprecision due to fuzziness has been recognized and applied in various aspects of geology such as stratigraphy [51], "fuzzy" kriging [52], evaluation of the provenance of glacial till [53], and "fuzzy" variograms [54], [55].

We present herein a new approach to dealing with fuzzy parameter estimation with special reference to the problems of predicting ore-grades within a mining deposit. Based on a collection of cluster-centers obtained from the fuzzy c-means algorithm, a fuzzy inference system is established to estimate grades located at these cluster centers. These cluster center-grade pairs act as the control information in the fuzzy space-grade system to infer unknown grades on the basis of fuzzy interpolation, fuzzy extrapolation, and fuzzy control. In the following sections, we introduce the concepts of fuzzy search domain, fuzzy rule base, and the criteria for fuzzy interpolation and fuzzy extrapolation for the proposed fuzzy grade estimator.

Determination of Fuzzy Search Domain

Based on the establishment of the cluster centers and the concepts of fuzzy sets, a fuzzy search domain Ω_i for calculating a grade within a fuzzy c-partition focused at v_i can be determined by the search-distance radius $r(\Omega_i)$, which is defined to be the closest distance from v_i to the nearest neighboring cluster center. The radius $r(\Omega_i)$ or r_i is expressed as

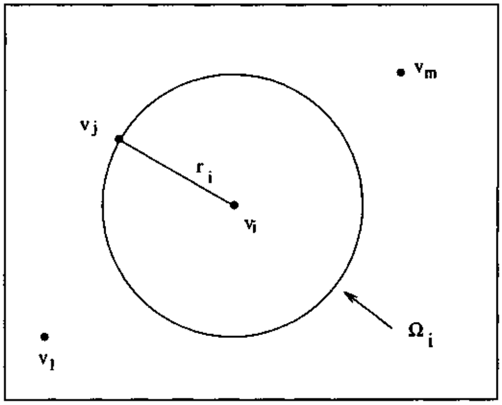
$$r_i = \min(d(v_i, v_k)). \tag{4.17}$$

Figure 4.12(a) illustrates the determination of fuzzy search domains.

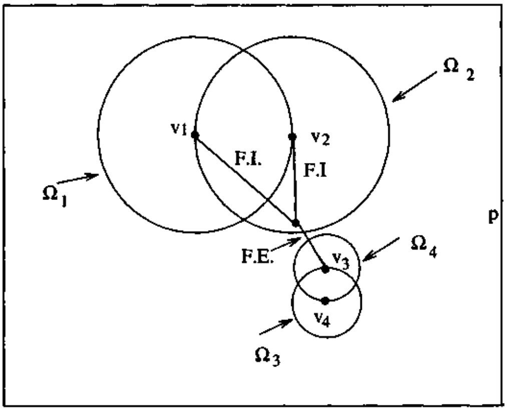
Fuzzy Interpolation and Fuzzy Extrapolation

Since the fuzzy search domains can be determined, a point p is defined to belong to Ω_i if the distance between p and v_i is not greater than r_i , that is

$$p \in \Omega_i \quad \text{if } d(p, v_i) \le r_i.$$



(a)



(b)

Figure 4.12 (a) Determination of a fuzzy search domain; (b) fuzzy interpolation (F.I.) and fuzzy extrapolation (F.E.).

Now the membership functions of p with respect to the cluster set V can be interpolated and/or extrapolated by the fuzzy-rule base (to be defined later) if it satisfies the following criteria:

$$\mu_p(V) = \begin{cases} \mu_p(v_i) & \text{if } p \in \Omega_i \text{ (fuzzy interpolation)} \\ \mu_p(v_j) & \text{if } p \in \Omega_i, p \notin \Omega_j \text{ and } d(p, v_i) \geq d(p, v_j) \\ & \text{(fuzzy extrapolation)} \\ 0 & \text{otherwise.} \end{cases} \tag{4.18}$$

In the case if p lies outside all the fuzzy domains, the computation for $\mu_p(V) \neq 0$ is allowed if $d(p, V) \leq \delta$ which is a given tolerant factor. Figure 4.12(b) shows the concepts of the proposed fuzzy interpolation and fuzzy extrapolation.

Grade Estimation of Cluster Centers

After the fuzzy domains, fuzzy interpolation and fuzzy extrapolation have been defined, the next step is to determine the grades of the cluster centers which will be subsequently used to compute the grade of any point in the fuzzy domains. In order to proceed with this process, it is necessary to set up a fuzzy control model, to construct the membership function $\mu_p(V)$, and to select a defuzzification method.

The structure of the knowledge base of a linguistic fuzzy model consists of a collection of a number of rules expressed in the form

$$\text{IF A is } X_i \text{ THEN B is } Y_i \ (i = 1, \cdots, m)$$

where X_i and Y_i are the fuzzy subsets of the input and output spaces and expressed in linguistic forms.

Based on the above expression, the relationship between the grades of the data points and the grades located at the cluster-centers can be modeled in a fuzzy linguistic form as follows:

IF P is *close* to V THEN g(V) is *close* to g(P)

where $\mathbf{P} = \{p_1, p_2, ..., p_n\}$ is the set of the data points, V is the set of the cluster centers, g(V) and g(P) are the sets of grades of V and P respectively, and "*close*" is a fuzzy set expressing the fuzzy relationship between the pairs (P,V) and (g(V), g(P)).

The membership function for the fuzzy term *close* can be assumed [56] to be a Gaussian-type function and is defined as:

$$\mu_P(V) = \exp\left[-0.5\left(\frac{P - V}{\sigma}\right)^2\right] \tag{4.19}$$

where σ is a constant which affects the shape of the Gaussian-type membership function.

For the data-point set P and the cluster-center set V as defined previously, the grade of a cluster center $v_i, g(v_i)$, can be obtained using a defuzzification scheme as follows:

$$g(v_i) = \frac{\sum_{j=1}^{n} \mu_{p_j}(v_i) g(p_j)}{\sum_{j=1}^{n} \mu_{p_j}(v_i)}. \tag{4.20}$$

The cluster-centers and their grades are now viewed as the coupled control points which are used to estimate the unknown grades as will be discussed below.

Fuzzy-control Grade at a Point

The grade at a particular location within its associated fuzzy search domains Ω_i $(i = 1, \cdots, k)$ is estimated by means of the defuzzification method as described in Eq. (4.20) with the associated control pairs. It is determined as

$$g(p_j) = \frac{\sum_{i=1}^{k} \mu_{p_j}(v_i) g(v_i)}{\sum_{i=1}^{k} \mu_{p_j}(v_i)} \tag{4.21}$$

where $g(p_j)$ is the estimated grade at a point in Ω_i, and $\mu_{p_j}(v_i)$ is the membership function of p in the domain of v_i, and is defined by

$$\mu_{p_j}(v_i) = \exp\left[-0.5\left(\frac{p_j - v_i}{\sigma}\right)^2\right]. \tag{4.22}$$

The following derivation is presented to explain why $g(v_i)$ is placed in Eq. (4.21).

Let $g(v_i)$ be denoted as g^*, which is set to be a control grade value within its domain Ω_i, and $\mu_{pk}(v_i) > 0$, k=n+1, be a membership grade of the unknown grade g_k at a point to be inferred. $\mu_{pj}(v_i)$, $\mu_{pk}(v_i)$, $g(p_j)$ and $g(p_k)$ are now shortly denoted as μ_j, μ_k, g_j and g_k respectively. Thus, the new sum of the membership grades becomes

$$\sum_{m=1}^{k} \mu_m = \left(\sum_{j=1}^{n} \mu_j\right) + \mu_k.$$

Based on Eq. (4.20) of the defuzzification scheme and keeping g^* unchanged (control point), g_k can be locally estimated as

$$g_k = \frac{\left(g^* \sum_{m=1}^{k} \mu_m\right) - \left(\sum_{j=1}^{n} \mu_j\, g_j\right)}{\mu_k}.$$

Substituting g^* defined by Eq. (4.20) in the above equation, it gives:

$$g_k = \frac{\sum_{j=1}^{n} \mu_j\, g_j\ \sum_{m=1}^{k} \mu_m}{\mu_k\ \sum_{j=1}^{n} \mu_j} - \frac{\sum_{j=1}^{n} \mu_j\, g_j}{\mu_k}$$

Table 4.2 Samples taken from a simulated iron ore deposit.

Easting	Northing	% Fe	Easting	Northing	% Fe
10	40	35.5	15	135	28.6
55	145	29.4	125	20	41.5
175	50	36.8	260	115	33.2
235	15	33.7	365	60	34.3
285	110	35.3	345	115	31.0
20	105	32.5	25	155	29.6
50	40	30.6	155	15	40.4
145	125	30.1	220	90	28.5
205	0	40.1	265	65	24.4
390	65	31.6	325	105	39.5
310	150	34.8			

or

$$g_k = \frac{\sum_{j=1}^{n} \mu_j\, g_j\, \sum_{m=1}^{k} \mu_m \; - \; \sum_{j=1}^{n} \mu_j\, \sum_{j=1}^{n} \mu_j g_j}{\mu_k \sum_{j=1}^{n} \mu_j}.$$

Finally, by some arrangements, g_k is found to be

$$g_k = \frac{g^*}{\mu_k}\left(\sum_{m=1}^{k} \mu_k - \sum_{j=1}^{n} \mu_j\right) = \frac{g^*}{\mu_k}(\mu_k) = g^*.$$

Estimation of an Iron Ore Grade

To illustrate the application of the proposed fuzzy algorithm to the problem of parameter identification in natural resources, Fig. 4.13(a) shows the example of an iron-ore deposit where the data set as shown in Table 4.2 is partially taken from [57]

Now we consider the point at the East-North (E-N) coordinates (145,125) with the iron grade of 30.1% as an unknown grade, and we want to estimate its grade. Figures 4.13(b) and 4.13(c) show the partitions of the clusters with three and four centers respectively, where the symbol "+" indicates the position of each cluster center. For the case of three cluster centers, the E-N coordinates of these centers obtained by the fuzzy c-means clustering method are: (29.47, 109.72), (322.61, 100.45) and (186.01, 30.23). For the case of four cluster centers, the coordinates of these are: (354.67, 86.55), (28.00, 114.51), (265.43, 99.31), and (166.02, 23.27). Table 4.3 shows the estimated % Fe using the fuzzy algorithm (the value of σ in Eq. (4.22) is taken to be 100), and the conventional inverse square distance method which is given by

$$g = \sum_{i=1}^{n} w_i g_i \tag{4.23}$$

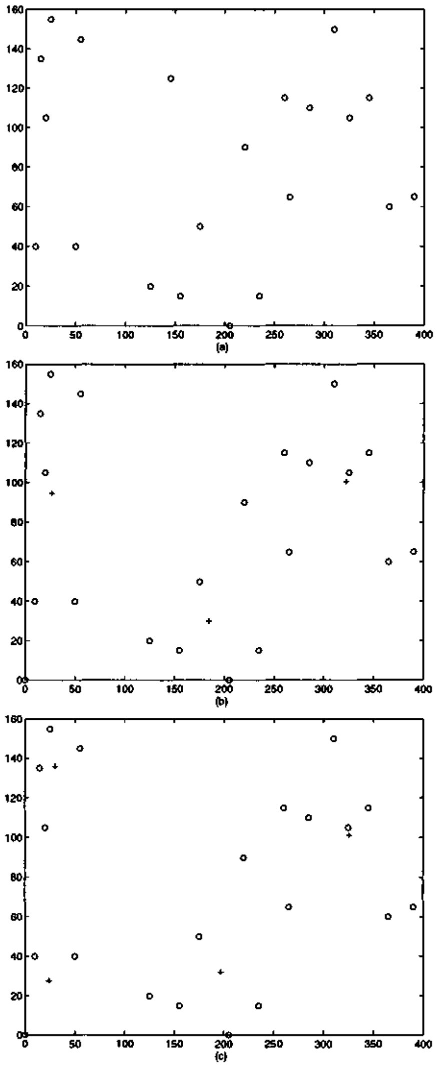


Figure 4.13 (a) An iron ore deposit; (b) partition with 3 clusters; and (c) partition with 4 clusters.

Table 4.3 Point estimates of iron grade.

Technique	Estimated %Fe	%Error
Fuzzy c-means (3 centers)	32.53	8.7
Fuzzy c-means (4 centers)	33.89	12.59
Inverse square distance	33.27	10.53

where g is the estimated grade, g_i is the grade of sample i, and w_i the weight assigned to sample i, and w_i is obtained by

$$w_i = \frac{f_i}{\sum_{i=1}^{n} f_i} \tag{4.24}$$

where $f_i = 1/(d_i)^2$, and d_i is the distance from sample i to the point of unknown grade.

The result obtained from the inverse square distance method is 33.27% with an error of 10.53%, whereas the results obtained from the proposed approach with three and four cluster centers are 32.53% and 33.89% with errors of 8.7% and 12.59% respectively. These results are also tabulated in Table 4.3. By observation, it is obvious that the case of three cluster centers is better than that of four cluster centers. The extra partition of the data set at the north-east location did not effectively contribute to the estimation as it is far away from the unknown parameter. For this problem, kriging is not applied here due to the insufficient amount of data for the construction of a semi-variogram.

4.7 Concluding Remarks

In this chapter, we have described the hard and fuzzy c-means clustering algorithms for unsupervised pattern classification. In both algorithms, the distance of an input sample to the center of the cluster which the input sample belongs to is used as a criterion to measure the cluster compactness. In the hard c-means algorithm, an input sample belongs to one cluster only, while in the fuzzy c-means algorithm the degree for which an input sample belongs to a cluster is represented by a membership value. We have made a comparison between the hard and fuzzy c-means algorithms for a document image segmentation problem and our experimental result shows that the fuzzy c-mean algorithm is more flexible and can produce a better segmentation result.

We have also described several cluster validity measures, including partition coefficient, partition entropy, and compactness and separation validity function. Our experiments show that the compactness and separation validity function gives the most accurate result for a color image segmentation problem.

A new approach to estimating fuzzy parameters with a particular application to the ore-grade problems has been discussed in this chapter. A fuzzy inference system and a fuzzy clustering algorithm have been applied as the underlying principles for this proposed approach. Starting with given data points, the fuzzy clustering method provides a collection of cluster centers, and unknown parameters can be estimated by a fuzzy model in a naturally consistent way. For the present application, the cluster centers and their estimated parameters are viewed as the control points in the fuzzy space-grade relationship, where the boundaries are not sharply defined, and are then used to infer a mean grade value at a particular location.

The inverse distance estimation has been found to be more sensible than the weighted mean [57]. However, the inverse distance weights are arbitrary as they solely depend on the distance between the samples and the unknown parameter, and will give less reliable results when the orientation or pattern of the samples are clustered. The point kriging estimator provides an optimal set of weights based on the optimality principle of a Lagrange multiplier, and the kriged weights change according to the geometrical arrangement of the samples. However, kriging methods may not give optimal solutions unless a semi-variogram which expresses the degree of spatial continuity of a regionalized variable is well defined. When the sampling points are inadequate for establishing a semi-variogram, kriging estimators are no longer reliable [54], [55]. For such a case of ill-defined problems, the use of the proposed method is more reasonable in both philosophical and practical aspects through the concept of the characteristic function of a fuzzy set, the optimal partition of sampling points using the fuzzy c-means clustering, and the natural and systematic framework of a fuzzy inference model.

Chapter 5

Line Pattern Matching

5.1 Introduction

Many pattern recognition problems can be simplified as a point or line pattern matching task [11], [12], [58], [59], [60]. For example, a Chinese character is composed of a number of strokes, each of which can be considered as a line segment, thus an input character can be recognized by matching its strokes against those of character prototypes [58], [59]. In this chapter, we consider fuzzy algorithms for matching line patterns. A point pattern can be converted to a line pattern by connecting the points with line segments, so line matching algorithms can be applied to point matching after a proper point to line transformation [11].

The matching problem we need to solve here can be stated as follows. We are given a set of pattern prototypes

$$P = \left\{p^{(1)}, p^{(2)}, \cdots, p^{(N_P)}\right\} \tag{5.1}$$

where N_P is the number of prototypes, and each prototype $p^{(k)}$ consists of M_k line segments

$$p^{(k)} = \left\{l_1^{(k)}, l_2^{(k)}, \cdots, l_{M_k}^{(k)}\right\} = \left\{(\mathbf{s}_1^{(k)}, \mathbf{e}_1^{(k)}), (\mathbf{s}_2^{(k)}, \mathbf{e}_2^{(k)}), \cdots, (\mathbf{s}_{M_k}^{(k)}, \mathbf{e}_{M_k}^{(k)})\right\} \tag{5.2}$$

where

$$l_j^{(k)} = (\mathbf{s}_j^{(k)}, \mathbf{e}_j^{(k)}) \tag{5.3}$$

represents the starting and ending locations respectively of line j in prototype k. An input pattern q consisting of N line segments is denoted as

$$q = \{l_1, l_2, \cdots, l_{M_k}\} = \{(\mathbf{s}_1, \mathbf{e}_1), (\mathbf{s}_2, \mathbf{e}_2), \cdots, (\mathbf{s}_N, \mathbf{e}_N)\} \tag{5.4}$$

where

$$l_j = (\mathbf{s}_j, \mathbf{e}_j) \tag{5.5}$$

represents the starting and ending locations respectively of line j in the input pattern. There are two classes of matching problems:

(1) $N_P = 1$, that is, there is only one prototype. In this case, we need to match two patterns p and q. The numbers of lines in the two patterns may not be equal, that is, one pattern may be a subset of the other. The task here is to determine the location of the sub-pattern in a larger pattern or to determine the correspondence relationship between the lines in the two patterns.

(2) $N_P > 1$. In this case, we need to find the prototype which is closest to the input pattern. We may also need to determine the correspondence relationship between the lines in the input pattern and the closest prototype.

For both problems, we can consider a prototype and the input pattern as two sets. In noise free cases, our task is to find a corresponding line in the larger set for each line in the smaller set. A straight forward method to solve this problem is to consider all possible correspondence relationships between the lines in the two sets. That is, we can try matching between any line in the smaller set and any line in the larger set. Suppose that the numbers of lines in the smaller and larger sets are N_1 and N_2 respectively and that one line on a set can only be matched to one line in the other set, then the matching procedure is equivalent to re-indexing of the lines in the larger set so that its first N_1 lines after re-indexing correspond to the lines in the smaller set. The total number of ways to re-index the larger set for this purpose is

$$N_2(N_2 - 1)(N_2 - 2) \cdots (N_2 - N_1 + 1) \geq N_1! \tag{5.6}$$

This means that it requires at least $N_1!$ evaluations to match a prototype and the input pattern. This is clearly unacceptable in practical applications.

As an example, we consider on-line recognition of character "king" shown in Fig. 5.1. The normal stroke order to write this character is shown in Fig. 5.1(a). However, one can produce the same character using the stroke order shown in Fig. 5.1(b) although children are taught not to write the character this way. For this character, there are $4! = 24$ different stroke order sequences. As the number of strokes increase, the number of possible stroke sequences increases rapidly. For example, character "queen" in Fig. 5.2(a) has 10 strokes. Now there are $10! = 3,628,800$ different stroke order sequences. It is obviously not feasible to implement a brute-force matching procedure that takes all stroke orders into account for all characters in a real system. Thus, most on-line handwritten Chinese character recognition systems often impose some constraints on the order of presentation of stroke lines in a character. An input pattern may be rejected without recognition or may be recognized incorrectly if one draws a stroke too early or too late. In some systems, the direction of writing may also be important. Normally, one draws a horizontal line from left to right and a vertical line from top to bottom when writing a Chinese character. If these rules are violated, a character may also be recognized incorrectly.

Line pattern matching can become more complicated if the input data contain noise or are distorted. In this case, there may be extra line segments or missing ones

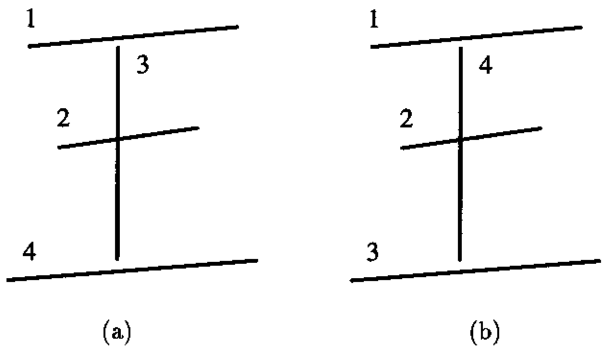


Figure 5.1 Chinese character "king". Children are taught to write this character according to the standard stroke order shown in (a), but one can write the same character according to the stroke order shown in (b). Altogether, the same character can be written in $4! = 24$ ways.

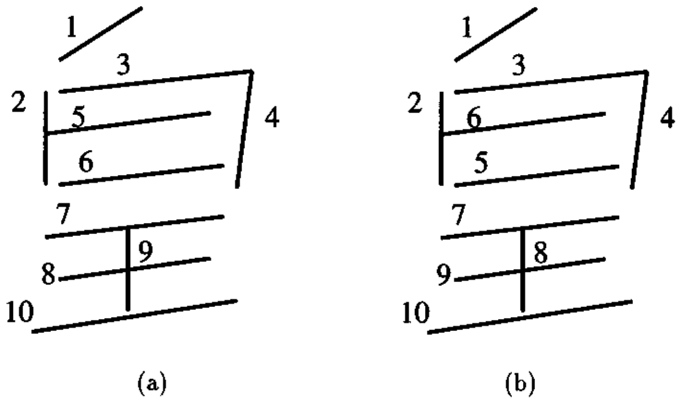


Figure 5.2 Chinese character "queen". The standard stroke order is shown in (a) and an alternative order is shown in (b). Altogether, the same character can be written in $10! = 3,628,800$ ways.

in the input pattern and the location and length of a line in the pattern may be different from those of the prototype. We need to take these problems into account to build a robust matching system.

Fuzzy algorithms can be used effectively for line pattern matching since they can deal with many ambiguous or fuzzy features of noisy line patterns. Cheng, Hsu and Chen have proposed a fuzzy method for handwritten Chinese character recognition [59]. Ogawa has published fuzzy relaxation based techniques for point pattern matching and partial shape matching [11], [12]. Algorithms described in this chapter are developed based on those originally proposed by Cheng, Hsu and Chen and by Ogawa. In Section 5.2, we describe how to use membership functions to measure similarities between line segments. In Section 5.3, we describe a basic algorithm for line pattern matching based on spatial relations of the lines in a prototype and the input pattern. In Sections 5.4 and 5.5, we consider more sophisticated algorithms to deal with noisy patterns and geometrically distorted patterns. Finally in Section 5.6, we show applications of line pattern matching algorithms to Chinese character recognition and point pattern matching.

5.2 Similarity Measures between Line Segments

The key problem in matching two line patterns is that for each line in one pattern we need to find the closest line in the other pattern. To carry out this task, we need to define how to measure the similarity between a pair of lines in two patterns. We consider three fuzzy sets, line location, line orientation, and directional relationship between lines and make use of the corresponding three membership functions for line similarity measurement.

5.2.1 Location Similarity

We can treat the mid-point of a line as its location. The distance between the locations of line i in prototype k and line j in the input pattern is shown in Fig. 5.3. and can be calculated as

$$d_{ij}^{(k)} = \left| \frac{\mathbf{s}_i^{(k)} + \mathbf{e}_i^{(k)}}{2D^{(k)}} - \frac{\mathbf{s}_j + \mathbf{e}_j}{2D} \right| \tag{5.7}$$

where $D^{(k)}$ and D are two normalization factors. In our experiments, we chose them to be the maximum width or maximum height, whichever is larger, of the prototype and the input pattern respectively. Assuming

$$\mathbf{s}_i^{(k)} = \begin{bmatrix} x_{si}^{(k)} \\ y_{si}^{(k)} \end{bmatrix}, \quad \mathbf{e}_i^{(k)} = \begin{bmatrix} x_{ei}^{(k)} \\ y_{ei}^{(k)} \end{bmatrix}, \quad \mathbf{s}_j = \begin{bmatrix} x_{sj} \\ y_{sj} \end{bmatrix}, \quad \mathbf{e}_j = \begin{bmatrix} x_{ej} \\ y_{ej} \end{bmatrix} \tag{5.8}$$

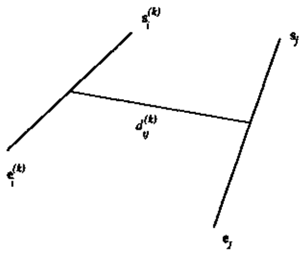


Figure 5.3 Distance between line i in prototype k and line j in the input pattern defined as the distance between the mid-points of the two lines.

we have

$$d_{ij}^{(k)} = \frac{1}{2}\sqrt{\left[\frac{x_{si}^{(k)} + x_{ei}^{(k)}}{D^{(k)}} - \frac{x_{sj} + x_{ej}}{D}\right]^2 + \left[\frac{y_{si}^{(k)} + y_{ei}^{(k)}}{D^{(k)}} - \frac{y_{sj} + y_{ej}}{D}\right]^2} \tag{5.9}$$

We can adjust the coordinate system so that

$$\min_i \left\{x_{si}^{(k)},\ x_{ei}^{(k)}\right\} = \min_i \left\{y_{si}^{(k)},\ y_{ei}^{(k)}\right\} = \min_j \{x_{sj},\ x_{ej}\} = \min_j \{y_{sj},\ y_{ej}\} = 0 \tag{5.10}$$

and

$$D^{(k)} = \max_i \left\{x_{si}^{(k)},\ x_{ei}^{(k)},\ y_{si}^{(k)},\ y_{ei}^{(k)}\right\},\quad D = \max_j \{x_{sj},\ x_{ej},\ y_{sj},\ y_{ej}\} \tag{5.11}$$

The location similarity between the two lines can be measured using the following membership function

$$\mu_{Lij}^{(k)} = \cos^{n_L}\left(c_L d_{ij}^{(k)}\right) \tag{5.12}$$

where n_L and c_L are constants. In our experiments, we chose $n_L = 3$ and $c_L = \pi/2$. For these parameters, the membership function is shown in Fig. 5.4. The minimum value of $d_{ij}^{(k)}$ is 0 and the maximum value is 1, corresponding to membership values of 1 and 0 respectively. In general, two lines have a large membership value if they are close to each other and a small membership value if they are far away from each other.

5.2.2 Orientation Similarity

The difference between the orientations of a pair of lines plays an important role in matching the two lines. We define the membership function to measure the orientation

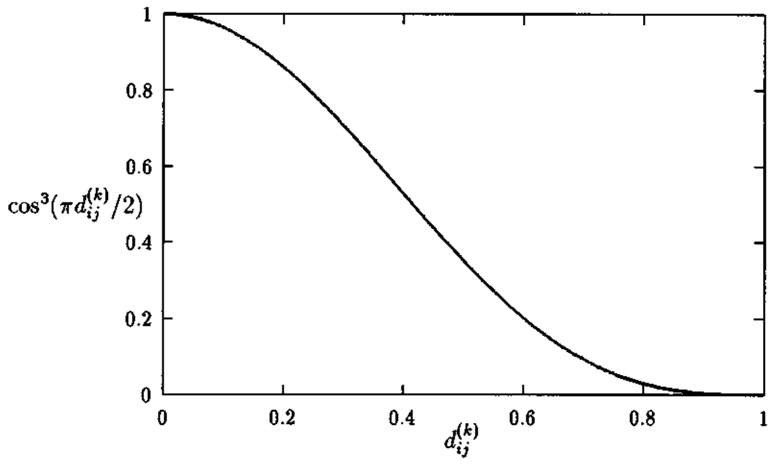


Figure 5.4 The membership function defined in Eq. (5.12) with $n_L = 3$ and $c_L = \pi/2$. This function is used for measuring the location similarity between line i in prototype k and line j in the input pattern.

similarity between line i in prototype k and line j in the input pattern in terms of the angle, $\theta_{ij}^{(k)}$, between the two lines as follows:

$$\mu_{Oij}^{(k)} = \left|\cos^{n_O}\left(c_O\theta_{ij}^{(k)}\right)\right| \tag{5.13}$$

where n_O and c_O are two constants, and

$$\theta_{ij}^{(k)} = \left|\arg(\mathbf{s}_i^{(k)} - \mathbf{e}_i^{(k)}) - \arg(\mathbf{s}_j - \mathbf{e}_j)\right| \tag{5.14}$$

where arg means the argument or phase of a vector. In our experiments, we chose $n_O = 3$, and $c_O = 1$.

In Fig. 5.5(a), the angle is measured anti-clockwise from line i in prototype k to line j in the input pattern. The membership value does not change if the angle is measured anti-clockwise from line j in the input pattern to line i in prototype k (see Fig. 5.5(b)). In general,

$$0 \leq \theta_{ij}^{(k)} \leq \pi \tag{5.15}$$

so the membership value is in the range of $[0, 1]$. Two lines have a large membership value if they have similar orientations and a small membership value if they have different orientations.

5.2.3 Relation Similarity

In many cases, it may be difficult to recognize a pattern by dealing with its components independently. The problem can be made easier by analyzing the relations

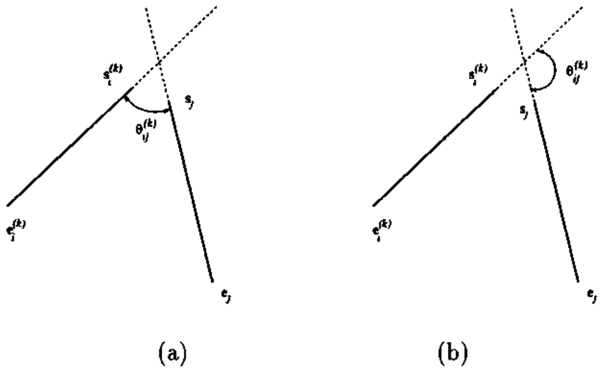


Figure 5.5 The angle between line i in prototype k and line j in the input pattern. This angle can be measured in two ways shown in (a) and (b) respectively. Both measures give the same value of the membership function defined in Eq. (5.14).

between the components. For line matching, we consider the directional relation between a pair of lines in a prototype and compare this relation with that in the input pattern. This means that we measure the similarity between a line in a prototype and a line in the input pattern based on evidence provided by other lines related to the two lines. For lines i and u in prototype k, we define the relation angle $\alpha_{iu}^{(k)}$ between the lines as the orientation of the line connecting the mid-points of the two lines (see Fig. 5.6(a)). For lines j and v in the input pattern, we define the relation angle α_{jv} between the two lines in the similar way (see Fig. 5.6(b)). The two relation angles are given by

$$\alpha_{iu}^{(k)} = \arg\left(\mathbf{s}_i^{(k)} + \mathbf{e}_i^{(k)} - \mathbf{s}_u^{(k)} - \mathbf{e}_u^{(k)}\right) \tag{5.16}$$

and

$$\alpha_{jv} = \arg\left(\mathbf{s}_j + \mathbf{e}_j - \mathbf{s}_v - \mathbf{e}_v\right) \tag{5.17}$$

In general,

$$0 \le \alpha_{iu}^{(k)},\ \alpha_{jv} \le 2\pi \tag{5.18}$$

We also have

$$\alpha_{ui}^{(k)} = 2\pi - \alpha_{iu}^{(k)} \tag{5.19}$$

and

$$\alpha_{vj} = 2\pi - \alpha_{jv} \tag{5.20}$$

This means that the starting line and ending line must be clearly defined to measure the relation angle between them (see Fig. 5.6). A relation angle shows a basic spatial relation between two lines. For example, if $\alpha_{iu}^{(k)} = \pi/2$, then we can say that line i is above line u or line u is below line i in prototype k.

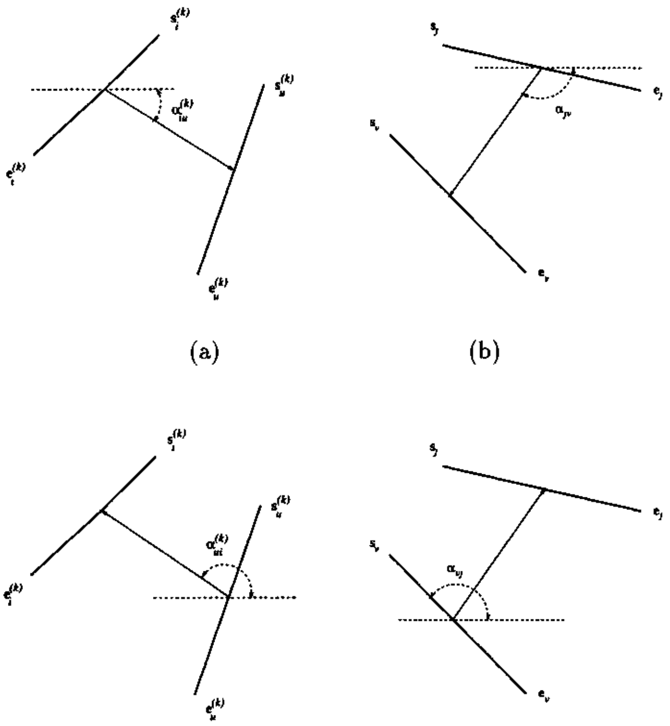


Figure 5.6 (a) The relation angle between lines i and u in prototype k, (b) the relation angle between lines j and v in the input pattern, (c) the relation angle between lines u and i in prototype k, and (d) the relation angle between lines v and j in the input pattern. Note that the starting line and the ending line must be clearly defined to measure a relation angle, that is, $\alpha_{iu}^{(k)} \neq \alpha_{ui}^{(k)}$ and $\alpha_{jv} \neq \alpha_{vj}$. In fact, $\alpha_{iu}^{(k)} + \alpha_{ui}^{(k)} = 2\pi$ and $\alpha_{jv} + \alpha_{vj} = 2\pi$.

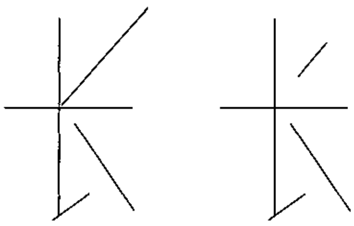


Figure 5.7 Two Chinese characters, both meaning "long". The two patterns should be recognized as the same character although the line in the upper right corner is much longer in one pattern than in the other pattern. In many cases, the length of a line in a Chinese character may vary widely without changing the meaning of the character.

We define the membership function for the relation similarity between lines i and u in prototype k and lines j and v in the input pattern as

$$\mu^{(k)}_{Riujv} = \left|\cos^{n_R}\left[c_R\left(\alpha^{(k)}_{iu} - \alpha_{jv}\right)\right]\right| \tag{5.21}$$

where n_R and c_R are two constants. In our experiments, we chose $n_R = 3$ and $c_R = 0.5$. The membership value is large if the relation angle between lines i and u in prototype k is close to that between lines j and v in the input pattern and the membership value is small if the angles have a large difference.

The relation similarity can be considered as a second order measure while location and orientation similarities are first order measures. When they are combined, lines in two patterns can be matched based on not only the properties of individual lines but also relations among these lines.

We do not define a fuzzy set for line lengths since their values for a given pattern can vary widely and cannot be represented accurately, such as in the case of handwritten characters shown in Fig. 5.7. However, in some other cases, the relative lengths of lines in a pattern can be important. For example, in Fig. 5.8, the character changes its meaning when the lower horizontal line becomes shorter than the upper one. This kind of problems can only be dealt with in practical applications based on special classification rules.

5.3 Basic Matching Algorithm

5.3.1 Entropy Measure

Our task here is to compare each prototype with the input pattern so that the correspondence relations between the lines in a prototype and the input pattern are identified and the best matching prototype is determined. We make use of three similarity measures described in the preceding section to match a pair of lines. To

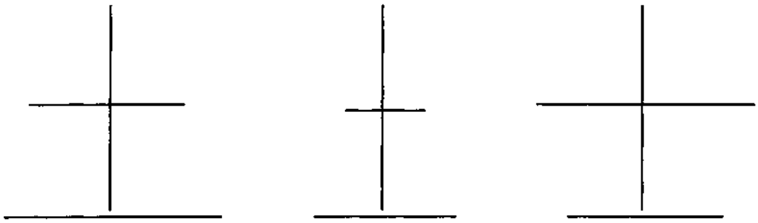


Figure 5.8 Three Chinese characters. The first two mean "soil" and the last one means "person". The absolute lengths of the horizontal lines are not important, but the relative lengths are important. In the second character, both horizontal lines are shorter than those in the first character, but the upper line is still shorter than the lower one, so the second character has the same meaning as the first one. However, in the third character, the upper horizontal line is longer than the lower one and it now represents a character with different meaning compared with the first and second characters.

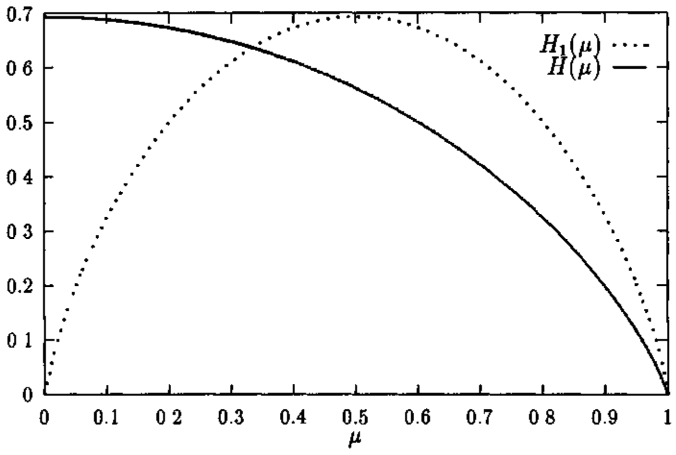


Figure 5.9 Entropy function $H_1(\mu) = -\mu \ln \mu - (1-\mu)\ln(1-\mu)$ and its modified version $H(\mu)$. $H_1(\mu)$ is monotonic decreasing only in the interval $0.5 \le \mu \le 1$. If we replace μ with $0.5 + 0.5\mu$ in $H_1(\mu)$, then the new function $H(\mu) = -(0.5+\mu)\ln(0.5 + 0.5\mu) - (0.5 - 0.5\mu)\ln(0.5 - 0.5\mu)$ is monotonic decreasing in the interval $0 \le \mu \le 1$, which is the same as the range where the membership values are distributed.

consider the matching cost for the entire pattern, we need to combine the similarity measures for all line pairs. This can be done using the Shannon's entropy function

$$H_1(\mu) = -\mu \ln \mu - (1-\mu)\ln(1-\mu) \tag{5.22}$$

This function is shown in Fig. 5.9. Since we need the matching cost to decrease as the similarity measure increases, we make use of only the second half of the entropy function, which is a monotonic decreasing function for $0.5 \leq \mu \leq 1$. Replacing μ with $0.5 + 0.5\mu$ in Eq. (5.22), we obtain

$$H(\mu) = -(0.5+0.5\mu)\ln(0.5+0.5\mu) - (0.5-0.5\mu)\ln(0.5-0.5\mu) \tag{5.23}$$

Now $H(\mu)$ is a monotonic decreasing function for $0 \leq \mu \leq 1$ (see Fig. 5.9), which can now be used as a cost function if we treat μ as the membership value of a similarity measure.

5.3.2 Matching Costs

We define the cost for matching line i in prototype k and line j in the input pattern in terms of location similarity as

$$C_{Lij}^{(k)} = -(0.5+0.5\mu_{Lij}^{(k)})\ln(0.5+0.5\mu_{Lij}^{(k)}) - (0.5-0.5\mu_{Lij}^{(k)})\ln(0.5-0.5\mu_{Lij}^{(k)}) \tag{5.24}$$

Similarly we define the cost for matching line i in prototype k and line j in the input pattern in terms of orientation similarity as

$$C_{Oij}^{(k)} = -(0.5+0.5\mu_{Oij}^{(k)})\ln(0.5+0.5\mu_{Oij}^{(k)}) - (0.5-0.5\mu_{Oij}^{(k)})\ln(0.5-0.5\mu_{Oij}^{(k)}) \tag{5.25}$$

It is more complicated to define the cost in terms of relation similarity. For line i in prototype k and line j in the input pattern, we consider line u in the prototype and line v in the input pattern, where

$$u = 1, 2, \cdots, i-1, i+1, \cdots, M_k \tag{5.26}$$

and

$$v = 1, 2, \cdots, j-1, j+1, \cdots, N \tag{5.27}$$

Equations (5.26) and (5.27) mean that all lines in the prototype and the input pattern except line i in the prototype and line j in the input pattern are considered. For the prototype, we can determine the relation similarity of line i with every other line corresponding to each u value. Similarly, for the input pattern, we can determine the relation similarity of line j with every other line corresponding to each j value. If line i in prototype k matches line j in the input pattern, then for each u value we can find a v value so that the relation between lines i and u in the prototype is similar to the relation between lines j and v in the input pattern. Now we define the cost for matching these relations as

$$\begin{aligned} C^{(k)}_{Riujv} &= -(0.5 + 0.5\mu^{(k)}_{Riujv}) \ln(0.5 + 0.5\mu^{(k)}_{Riujv}) \\ &\quad -(0.5 - 0.5\mu^{(k)}_{Riujv}) \ln(0.5 - 0.5\mu^{(k)}_{Riujv}) \end{aligned} \tag{5.28}$$

For each u value, we can find a v value so that $C^{(k)}_{Riujv}$ is a minimum. The cost for matching line i in prototype k and line j in the input pattern in terms of relation similarity can now be defined as

$$C^{(k)}_{Rij} = \frac{1}{2(M_k - 1)} \sum_{1 \le u \le M_k, u \ne i} \min_{1 \le v \le N, v \ne j} \left(C^{(k)}_{Ouv} + C^{(k)}_{Riujv} \right) \tag{5.29}$$

The cost for matching line i in prototype k and line j in the input pattern can be defined as the weighted summation of the three components corresponding to three similarity measures:

$$C^{(k)}_{ij} = a_L C^{(k)}_{Lij} + a_O C^{(k)}_{Oij} + a_R C^{(k)}_{Rij} \tag{5.30}$$

where a_L, a_O and a_R are weighting constants. We choose $a_L = a_O = a_R = 1$ in our experiments. In general, these constants can be adjusted according to training data to achieve the best matching accuracy. Finally, the cost for matching prototype k and the input pattern is

$$C^{(k)} = \frac{1}{M_k} \sum_{i=1}^{M_k} \min_{j=1}^{N} C^{(k)}_{ij} \tag{5.31}$$

That is, for each line in a prototype, we find the best matching line in the input pattern and then obtain the cost of matching the prototype and the input pattern by averaging the costs associated with all lines in the prototype. The input pattern can then be assigned to the class of the prototype for which the matching cost is minimum, that is,

$$\text{class}(q) = \text{class}(p^{(k^*)}) \tag{5.32}$$

where class$(\cdot)$ means the class label of a pattern and

$$k^* = \arg \min_{k=1}^{M_k} C^{(k)} \tag{5.33}$$

An input pattern can be classified based on $C^{(k)}$. To determine the correspondence relations between the lines in the best matching prototype $p^{(k^*)}$ and the input pattern q, we can make use of the cost function $C^{(k)}_{ij}$. The line in the input pattern that best matches line i in the prototype is

$$j_i = \arg \min_{j=1}^{N} C^{(k)}_{ij} \tag{5.34}$$

5.3.3 Examples

We consider prototype patterns in Figs. 5.10(a) and 5.10(b) and input patterns in Fig. 5.11(a) to 11(d). The matching cost $C^{(k)}$ is shown in Table 5.1, where $k = 1$ corresponds to the prototype in Fig. 5.10(a) and $k = 2$ to the prototype in Fig. 5.10(b). The pattern matching costs are:

Prototype \ Input	Fig. 5.11(*a*)	Fig. 5.11(*b*)	Fig. 5.11(*c*)	Fig. 5.11(*d*)
Fig. 5.10(*a*) $(k = 1)$	0.129	0.591	0.129	0.591
Fig. 5.10(*b*) $(k = 2)$	0.671	0.159	0.671	0.159

The matching algorithm has worked reliably for these patterns. It shows the following properties:

- Patterns in the same class are matched with low cost.
- Patterns in different classes are matched with high cost.
- Patterns in the same class are matched correctly despite changes in line labels or the stroke order.

$C_{ij}^{(k)}$ is shown in Tables 5.1 to 5.6 for different combinations of prototypes and input patterns. For each line in a prototype, we can determine the best matching line in the input pattern by searching for the minimum match cost along each row in the matching cost matrix. In these tables, the best matching lines are shown under the j_i columns and the corresponding line matching costs are shown under the $C_{ij_i}^{(k)}$ columns. To evaluate the reliability of line matching, we define the relative line matching cost for line i in prototype k and the best matching line in the input pattern, line j_i, as

$$L_{ij_i}^{(k)} = \frac{C_{ij_i}^{(k)}}{\max\limits_{1\le j\le M_k} C_{ij}^{(k)} - \min\limits_{1\le j\le M_k} C_{ij}^{(k)}} = \frac{C_{ij_i}^{(k)}}{\max\limits_{1\le j\le M_k} C_{ij}^{(k)} - C_{ij_i}^{(k)}} \tag{5.35}$$

The last columns of Tables 5.1 to 5.6 show this relative line matching cost for our examples. $L_{ij_i}^{(k)}$ varies in a wider range than $C_{ij_i}^{(k)}$, that is, it is smaller than $C_{ij_i}^{(k)}$ when the matching cost is small and larger than $C_{ij_i}^{(k)}$ when the matching cost is large.

From Tables 5.1 to 5.6, we can see that the relative line matching cost for every line is small when a prototype and the input pattern belong to the same class, no matter how the lines in the input pattern are labeled. However, when a prototype and the input pattern belong to different classes, the relative matching cost for most lines is large. For example, if we set a threshold $T = 0.30$ for $L_{ij_i}^{(k)}$, then all lines in Tables 5.1 and 5.4 to 5.6, but only one line in Tables 5.2 or 5.3, can be considered matched reliably.

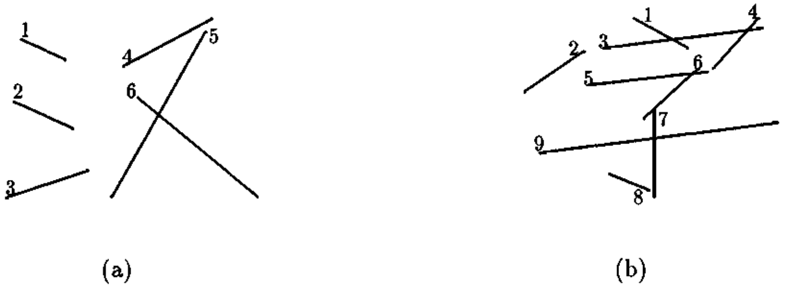


Figure 5.10 Examples of Chinese characters. These two line patterns are used as prototypes in the experiment.

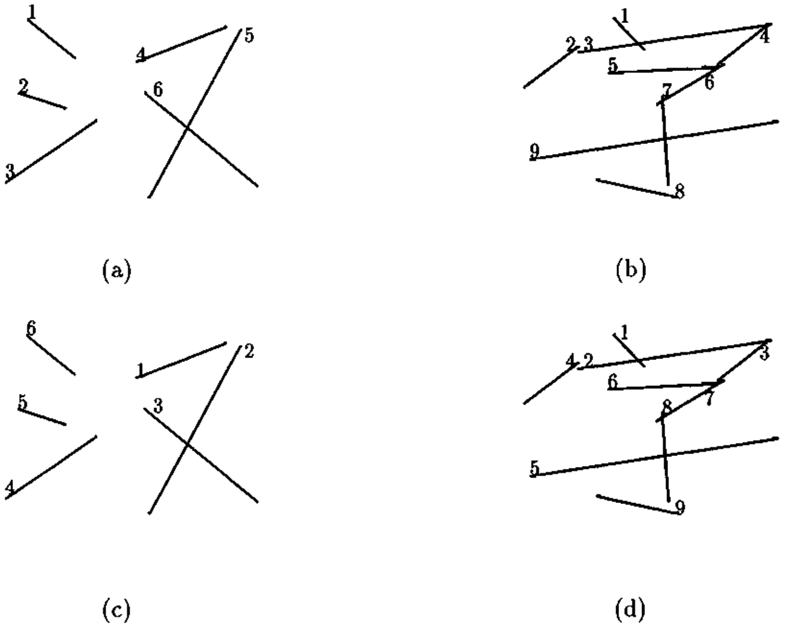


Figure 5.11 Chinese characters used as input samples in the experiment. Characters in (a) and (c) are the same, but the stroke orders or the line labels in two characters are different. Similarly characters in (b) and (d) are the same and their line labels are different.

Table 5.1 Line matching cost matrix $C_{ij}^{(1)}$ for the prototype in Fig. 5.10(a) and the input character in Fig. 5.11(a).

i	$C_{i1}^{(k)}$	$C_{i2}^{(k)}$	$C_{i3}^{(k)}$	$C_{i4}^{(k)}$	$C_{i5}^{(k)}$	$C_{i6}^{(k)}$	j_i	$C_{ij_i}^{(k)}$	$L_{ij_i}^{(k)}$
1	0.13	0.46	1.55	1.54	1.69	1.36	1	0.13	0.09
2	0.59	0.07	1.06	1.72	1.69	1.17	2	0.07	0.04
3	1.69	1.12	0.17	1.21	1.38	1.64	3	0.17	0.11
4	1.42	1.73	1.30	0.11	1.01	1.82	4	0.11	0.06
5	1.59	1.50	1.02	0.86	0.21	1.07	5	0.21	0.15
6	1.29	1.24	1.69	1.51	0.79	0.08	6	0.08	0.05

Table 5.2 Line matching cost matrix $C_{ij}^{(1)}$ for the prototype in Fig. 5.10(a) and the input character in Fig. 5.11(b).

i	$C_{i1}^{(k)}$	$C_{i2}^{(k)}$	$C_{i3}^{(k)}$	$C_{i4}^{(k)}$	$C_{i5}^{(k)}$	$C_{i6}^{(k)}$	$C_{i7}^{(k)}$	$C_{i8}^{(k)}$	$C_{i9}^{(k)}$	j_i	$C_{ij_i}^{(k)}$	$L_{ij_i}^{(k)}$
1	0.51	1.03	1.11	1.49	1.21	1.43	1.59	1.33	1.69	1	0.51	0.43
2	0.92	0.96	1.40	1.71	1.17	1.30	1.44	1.07	1.54	1	0.92	1.17
3	1.75	1.03	1.13	1.61	0.81	1.03	1.11	1.40	0.45	9	0.45	0.35
4	1.27	1.14	0.47	0.45	0.80	0.77	1.61	2.22	1.15	4	0.45	0.25
5	1.41	1.05	1.08	0.52	1.03	0.48	1.16	1.90	1.34	6	0.48	0.33
6	1.37	1.89	1.70	1.36	1.47	1.26	1.28	1.18	1.61	8	1.18	1.68

Table 5.3 Line matching cost matrix $C_{ij}^{(2)}$ for the prototype in Fig. 5.10(b) and the input character in Fig. 5.11(a).

i	$C_{i1}^{(k)}$	$C_{i2}^{(k)}$	$C_{i3}^{(k)}$	$C_{i4}^{(k)}$	$C_{i5}^{(k)}$	$C_{i6}^{(k)}$	j_i	$C_{ij_i}^{(k)}$	$L_{ij_i}^{(k)}$
1	0.48	0.85	1.80	1.19	1.53	1.13	1	0.48	0.36
2	1.07	0.93	0.58	1.24	1.27	1.93	3	0.58	0.43
3	1.39	1.51	1.35	0.35	1.18	1.65	4	0.35	0.27
4	1.64	1.76	1.49	0.56	0.55	1.43	5	0.55	0.46
5	1.40	1.27	0.96	0.89	1.25	1.65	4	0.89	1.18
6	1.59	1.45	0.92	1.15	0.60	1.36	5	0.60	0.61
7	1.57	1.47	1.07	1.88	0.91	1.09	5	0.91	0.95
8	1.22	0.96	1.30	2.12	1.56	0.82	6	0.82	0.63
9	1.79	1.55	0.85	1.25	1.14	1.37	3	0.85	0.90

Table 5.4 Line matching cost matrix $C_{ij}^{(2)}$ for the prototype in Fig. 5.10(b) and the input character in Fig. 5.11(b).

i	$C_{i1}^{(k)}$	$C_{i2}^{(k)}$	$C_{i3}^{(k)}$	$C_{i4}^{(k)}$	$C_{i5}^{(k)}$	$C_{i6}^{(k)}$	$C_{i7}^{(k)}$	$C_{i8}^{(k)}$	$C_{i9}^{(k)}$	j_i	$C_{ij_i}^{(k)}$	$L_{ij_i}^{(k)}$
1	0.39	1.33	0.78	0.96	1.08	1.20	1.59	1.40	1.82	1	0.39	0.28
2	1.05	0.10	0.91	1.12	0.90	0.78	1.39	1.83	1.07	2	0.10	0.06
3	1.14	1.29	0.17	0.58	0.45	0.85	1.78	2.03	1.15	3	0.17	0.09
4	1.27	1.12	0.90	0.15	1.18	0.67	1.41	2.03	1.59	4	0.15	0.08
5	1.24	1.02	0.51	1.01	0.08	0.55	1.38	1.78	0.75	5	0.08	0.05
6	1.33	0.82	0.98	0.56	0.62	0.07	0.92	1.69	0.91	6	0.07	0.05
7	1.27	1.17	1.74	1.30	1.25	0.82	0.08	1.05	0.71	7	0.08	0.05
8	1.42	1.57	1.93	1.88	1.46	1.37	0.75	0.17	0.93	8	0.17	0.09
9	1.70	1.28	1.02	1.36	0.52	0.70	0.81	1.38	0.21	9	0.21	0.14

Table 5.5 Line matching cost matrix $C_{ij}^{(1)}$ for the prototype in Fig. 5.10(a) and the input character in Fig. 5.11(c).

i	$C_{i1}^{(k)}$	$C_{i2}^{(k)}$	$C_{i3}^{(k)}$	$C_{i4}^{(k)}$	$C_{i5}^{(k)}$	$C_{i6}^{(k)}$	j_i	$C_{ij_i}^{(k)}$	$L_{ij_i}^{(k)}$
1	1.54	1.69	1.36	1.55	0.46	0.13	6	0.13	0.09
2	1.72	1.69	1.17	1.06	0.07	0.59	5	0.07	0.04
3	1.21	1.38	1.64	0.17	1.12	1.69	4	0.17	0.11
4	0.11	1.01	1.82	1.30	1.73	1.42	1	0.11	0.06
5	0.86	0.21	1.07	1.02	1.50	1.59	2	0.21	0.15
6	1.51	0.79	0.08	1.69	1.24	1.29	3	0.08	0.05

Table 5.6 Line matching cost matrix $C_{ij}^{(2)}$ for the prototype in Fig. 5.10(b) and the input character in Fig. 5.11(d).

i	$C_{i1}^{(k)}$	$C_{i2}^{(k)}$	$C_{i3}^{(k)}$	$C_{i4}^{(k)}$	$C_{i5}^{(k)}$	$C_{i6}^{(k)}$	$C_{i7}^{(k)}$	$C_{i8}^{(k)}$	$C_{i9}^{(k)}$	j_i	$C_{ij_i}^{(k)}$	$L_{ij_i}^{(k)}$
1	0.39	0.78	0.96	1.33	1.82	1.08	1.20	1.59	1.40	1	0.39	0.28
2	1.05	0.91	1.12	0.10	1.07	0.90	0.78	1.39	1.83	4	0.10	0.06
3	1.14	0.17	0.58	1.29	1.15	0.45	0.85	1.78	2.03	2	0.17	0.09
4	1.27	0.90	0.15	1.12	1.59	1.18	0.67	1.41	2.03	3	0.15	0.08
5	1.24	0.51	1.01	1.02	0.75	0.08	0.55	1.38	1.78	6	0.08	0.05
6	1.33	0.98	0.56	0.82	0.91	0.62	0.07	0.92	1.69	7	0.07	0.05
7	1.27	1.74	1.30	1.17	0.71	1.25	0.82	0.08	1.05	8	0.08	0.05
8	1.42	1.93	1.88	1.57	0.93	1.46	1.37	0.75	0.17	9	0.17	0.09
9	1.70	1.02	1.36	1.28	0.21	0.52	0.70	0.81	1.38	5	0.21	0.14

5.4 Dealing with Noisy Patterns

From the examples in Section 5.3, we know that $L_{ij_i}^{(k)}$ provides useful information not only for the line matching relations, but also the overall similarity between a prototype and the input pattern. If all lines in a prototype are matched with all lines in the input pattern with low $L_{ij_i}^{(k)}$, then we can safely say that the input pattern belongs to the class of the prototype. On the other hand, if only a very small number of lines in a prototype are matched with a small number of lines in the input pattern with low $L_{ij_i}^{(k)}$, then we can safely say that the input pattern does not belong to the class of the prototype. In practical applications, however, a pattern may have missing or extra lines because of noise corruption, so we do not expect that every line in a prototype to be matched with a line in the input pattern with a low cost even if the two patterns belong to the same class. That is, we will have cases in which a small number of lines in a prototype are not matched well with the input pattern. We show below that we can identify the noisy lines or missing lines in the input pattern based on analysis of $L_{ij_i}^{(k)}$. We deal with prototypes in Figs. 5.10(a) and 5.10(b) and noisy characters in Figs. 5.12(a) to 5.12(d) to explain the techniques.

First, we need to determine the best matching prototype. This can be carried out with the basic matching algorithm described in the preceding section. For the two prototypes, the pattern matching costs are:

Prototype \ Input	Fig. 5.12(a)	Fig. 5.12(b)	Fig. 5.12(c)	Fig. 5.12(d)
Fig. 5.10(a) ($k = 1$)	0.122	0.122	0.267	0.260
Fig. 5.10(b) ($k = 2$)	0.543	0.516	0.674	0.546

Obviously, all input patterns belong to the class of the first prototype.

In the following subsections, we consider the first prototype only and describe the procedures for dealing with noisy and missing lines in an input pattern.

5.4.1 Detecting and Removing Extra Lines

Consider the input pattern in Fig. 5.12(a). It has an extra horizontal line in the middle of the character. When the line pattern in Fig. 5.10(a) is used as a prototype, the best matching line labels and the relative line matching costs are:

i	1	2	3	4	5	6
j_i	1	2	3	4	5	6
$L_{ij_i}^{(k)}$	0.08	0.05	0.16	0.05	0.08	0.06

We can see that lines 1 to 6 in the prototype are matched with lines 1 to 6 in the input pattern with low cost. Line 7, the noisy line, in the input pattern is not matched with any line in the prototype, so we can now safely say that the input pattern belongs to the class of the prototype if line 7 is ignored. This result can be verified by matching

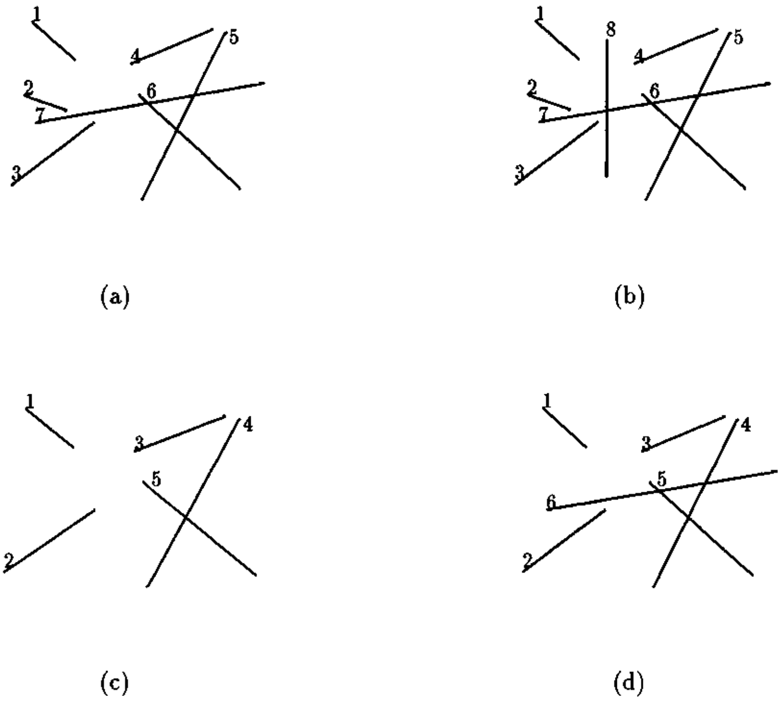


Figure 5.12 Characters containing extra or missing strokes. Information provided by the relative line matching costs can be used to deal with these noisy patterns.

the two patterns again with the prototype in Fig. 5.10(a) treated as the input pattern and the input pattern in Fig. 5.12(a) as a prototype. That is, we exchange the roles of the two patterns. Now the best matching line labels and the relative line matching costs are:

i	1	2	3	4	5	6	7
j_i	1	2	3	4	5	6	5
$L^{(k)}_{ij_i}$	0.11	0.07	0.19	0.14	0.11	0.07	0.70

We can see that both line 5 and line 7 in the prototype (the original input pattern) match line 5 in the input pattern (the original prototype) with relative line matching costs equal to 0.11 and 0.70 respectively. Since line 5 in the prototype matches line 5 in the input pattern with a much lower cost than line 7 in the prototype, line 7 in the prototype must be an extra line which does not match any line in the input pattern. So, in both matching procedures we can identify the noisy line in the original input pattern.

Now we consider the input pattern in Fig. 5.12(b) which contains two noisy lines. When the line pattern in Fig. 5.10(a) is used as a prototype, the best matching line labels and the relative line matching costs are:

i	1	2	3	4	5	6
j_i	1	2	3	4	5	6
$L^{(k)}_{ij_i}$	0.08	0.05	0.16	0.05	0.08	0.06

When the input pattern is treated as a prototype and the line pattern in Fig. 5.10(a) is used as the input pattern, the best matching line labels and the relative line matching costs are:

i	1	2	3	4	5	6	7	8
j_i	1	2	3	4	5	6	5	8
$L^{(k)}_{ij_i}$	0.12	0.08	0.20	0.15	0.15	0.08	0.78	0.85

From either of the matching procedure, we can determine that lines 7 and 8 in the original input pattern are noisy lines.

5.4.2 Detecting and Recovering Missing Lines

Consider the character in Fig. 5.12(c), which has a line missing. When this character is used as the input pattern and the line pattern in Fig. 5.10(a) is used as a prototype, the best matching line labels and the relative line matching costs are:

i	1	2	3	4	5	6
j_i	1	1	2	3	4	5
$L^{(k)}_{ij_i}$	0.12	0.52	0.14	0.17	0.18	0.07

Note that $L_{ij_i}^{(k)}$ is higher than that in Table 5.1 since there is a line missing in the input pattern so that the relation similarity between the prototype and the input pattern is reduced. However, for most lines the relative line matching cost is still fairly low. For both $i = 1$ and $i = 2$, we have $j_i = 1$. This means that two lines in the prototype match one line in the input pattern. To resolve the conflict, we check the values of $L_{ij_i}^{(k)}$. Since $L_{1j_1}^{(k)}$ is smaller than $L_{2j_2}^{(k)}$, line 1 and not line 2 in the prototype matches line 1 in the input pattern. This also means line 2 in the prototype does not have a partner or the corresponding line is missing in the input pattern. This result can also be verified by a second matching procedure. When the input pattern is treated as a prototype and the line pattern in Fig. 5.10(a) is used as the input pattern, the best matching line labels and the relative line matching costs are:

i	1	2	3	4	5
j_i	1	3	4	5	6
$L_{ij_i}^{(k)}$	0.09	0.12	0.20	0.15	0.06

We can see that line 2 in the input pattern (the original prototype) is not in the list and thus its partner in the prototype (the original input pattern) is missing. So from either of the matching procedures, we can detect and recover the missing line.

Now we consider the character in Fig. 5.12(d). It has a missing line and an extra line. When this character is used as the input pattern and the line pattern in Fig. 5.10(a) is used as a prototype, the best matching line labels and the relative line matching costs are:

i	1	2	3	4	5	6
j_i	1	1	2	3	4	5
$L_{ij_i}^{(k)}$	0.12	0.57	0.18	0.16	0.10	0.08

Similar to the character in Fig. 5.12(c), line 1 in the input pattern should match line 1 and not line 2 in the prototype, based on the corresponding values of $L_{ij_i}^{(k)}$. Thus, line 2 in the prototype does not have a partner and its corresponding line is missing in the input pattern. Line 6 does not match any line in the prototype, so it is an extra line. When the input pattern is treated as a prototype and the line pattern in Fig. 5.10(a) is used as the input pattern, the best matching line labels and the relative line matching costs are:

i	1	2	3	4	5	6
j_i	1	3	4	5	6	5
$L_{ij_i}^{(k)}$	0.12	0.19	0.17	0.13	0.08	0.82

Line 2 in the input pattern (the original prototype) is missing in the above list, so the corresponding line in the prototype (the original input pattern) is missing. Line 5 in the input pattern matches lines 4 and 6 in the prototype with relative line matching costs 0.13 and 0.82 respectively. Since $L_{6j_6}^{(k)}$ is large, line 6 in the prototype does not have a partner in the input pattern and this line in the prototype (the original input pattern) should be considered as an extra line.

5.4.3 Summary of Matching Algorithms

In summary, we can use the following rules to detect missing and extra lines in an input pattern, based on the information of relative line matching costs between a prototype and the input pattern:

(1) Select the best matching prototype and calculate the relative line matching cost for each line in the prototype.

(2) Choose a threshold T for the relative line matching costs and count the number of lines in the prototype with relative line matching costs below threshold T. We call this number M_T.

(3) Choose a threshold T_M. If $M_T < T_M$, reject the pattern without further processing, otherwise go to the next step. Steps 2 and 3 ensure that the input pattern indeed belongs to the class of the best matching prototype. If there are not enough lines in the input pattern which can match the prototype with low cost, the input pattern cannot be considered as being in the same class as the prototype and is simply rejected.

(4) For all i values (line labels of the prototype), check j_i and $L_{ij_i}^{(k)}$ values. If j_i is the same for several i values, delete all j_i entries except the one with the lowest $L_{ij_i}^{(k)}$ value.

(5) Check each j_i entry. If it is empty (its value is deleted in step 4), then line i in the prototype does not have a partner and its corresponding line is missing in the input pattern.

(6) Check the label of each line in the input pattern. If it does not appear in any of the j_i entries, then the line is an extra line, that is, it should not exist when the input pattern is considered as being in the same class of the prototype.

5.5 Dealing with Rotated Patterns

5.5.1 Basic Algorithm

Consider rotated characters in Fig. 5.13. A brute-force approach to matching rotated patterns is to treat the rotation angle as a variable and match the prototypes

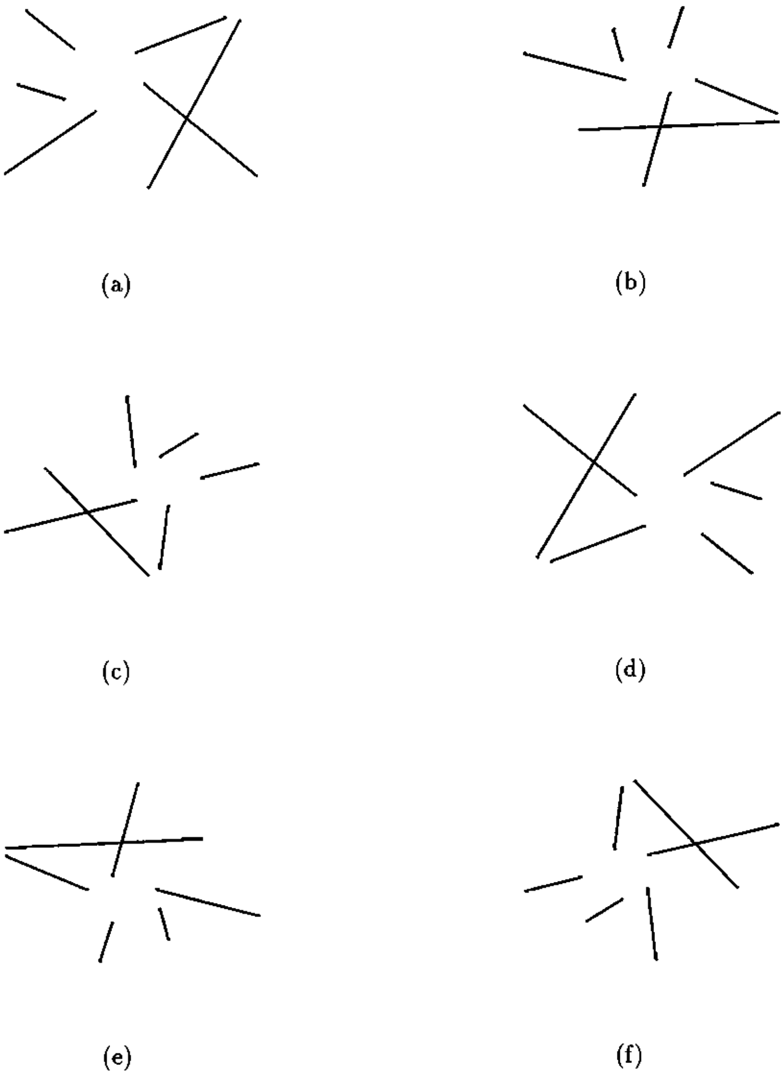


Figure 5.13 A Chinese character and its rotated versions: (a) the original character, (b) the character is rotated by -60^0, (c) the character is rotated by -120^0, (d) the character is rotated by -180^0, (e) the character is rotated by -240^0, and (f) the character is rotated by -300^0.

with all possible rotated versions of the input pattern. We define a new matching cost function

$$C_\beta^{(k)} = \frac{1}{M_k} \sum_{i=1}^{M_k} \min_{j=1}^{N} C_{\beta ij}^{(k)} \tag{5.36}$$

where β is the angle of rotation of the input pattern and

$$C_{\beta ij}^{(k)} = a_L C_{\beta Lij}^{(k)} + a_O C_{\beta Oij}^{(k)} + a_R C_{\beta Rij}^{(k)} \tag{5.37}$$

where three matching cost components $C_{\beta Lij}^{(k)}$, $C_{\beta Oij}^{(k)}$, and $C_{\beta Rij}^{(k)}$ are calculated between prototype k and the input pattern rotated through an angle β. The corresponding algorithm has the following steps:

(1) Choose $\Delta\beta$, the increment between two rotation angles.

(2) Set $\beta = 0$.

(3) Rotate the input pattern through the angle β.

(4) Calculate $C_\beta^{(k)}$ for all k values.

(5) Set $\beta = \beta + \Delta\beta$.

(6) If $\beta < 360^0$, go to step (3).

(7) For all β values and k values, determine the minimum $C_\beta^{(k)}$.

For the rotated pattern in Fig. 5.13(b) and two prototypes in Figs. 5.10(a) and 5.10(b), the matching cost functions are calculated for $\beta = 0^0, 1^0, 2^0, \cdots, 359^0$ and are shown in Fig. 5.14. We can see that $C_\beta^{(k)}$ is minimum for $k = 1$ and $\beta = 60^0$. This means that the original pattern was rotated by -60^0 compared with the orientation of the best matching prototype.

5.5.2 Fast Algorithm

The basic algorithm for matching rotated patterns is time consuming. In steps (3) and (4), we have to rotate the input pattern for every β value, normalize the pattern, compute the membership functions, and calculate the matching cost. There is not simple relation between the location similarities measured between a prototype and the original and the rotated input patterns. However, the orientation of a line and the relation angle between two lines in the input pattern are simply increased or decreased by β when the entire pattern is rotated by an angle β. Thus, we can calculate $C_{\beta Oij}^{(k)}$ and $C_{\beta Rij}^{(k)}$ based on information of the original input pattern without actually rotating it. Now we define a new cost function

$$C'^{(k)}_\beta = \frac{1}{M_k} \sum_{i=1}^{M_k} \min_{j=1}^{N} C'^{(k)}_{\beta ij} \tag{5.38}$$

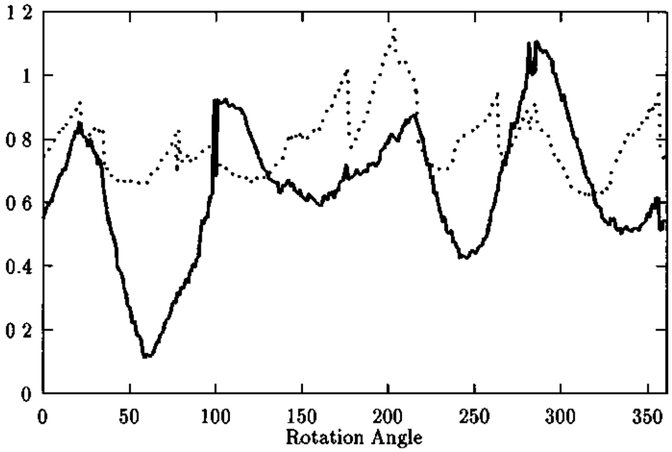


Figure 5.14 Pattern matching cost functions, $C_\beta^{(1)}$ (solid line) and $C_\beta^{(2)}$ (dotted line), for the rotated pattern in Fig. 5.13(b) and two prototypes in Figs. 5.10(a) and 5.10(b) respectively.

where

$$C'^{(k)}_{\beta ij} = a_O C^{(k)}_{\beta Oij} + a_R C^{(k)}_{\beta Rij} \tag{5.39}$$

Since $C'^{(k)}_\beta$ does not contain the term $C^{(k)}_{\beta Lij}$ that requires additional computations, it can be calculated easily for different β values by simply changing the $\theta^{(k)}_{ij}$ and $\alpha^{(k)}_{iujv}$ values.

For rotated patterns in Fig. 5.13, the cost functions $C'^{(k)}_\beta$ $(k = 1, 2)$ are calculated for $\beta = 0^0, 1^0, 2^0, \cdots, 359^0$ and are shown in Fig. 5.15. The cost functions are minimum for $k = 1$ and β equal to 0^0, 60^0, 120^0, 180^0, 240^0, and 300^0 respectively, corresponding to the rotation angles of the original input pattern.

After we find the value of β for which $C'^{(k)}_\beta$ is minimum, we can correct the rotation of the input pattern and deal with the rotated pattern based on the algorithms described in Sections 5.3 and 5.4. In fact, we can even obtain correct classification based on $C'^{(k)}_\beta$. However, it is not as reliable as $C^{(k)}$ since it does not contain the position information. To ensure the matching reliability, we can find several k and β values for which $C'^{(k)}_\beta$ is locally minimum, rotate the pattern for all the β values and then calculate $C^{(k)}$ for all the β values to classify the original pattern.

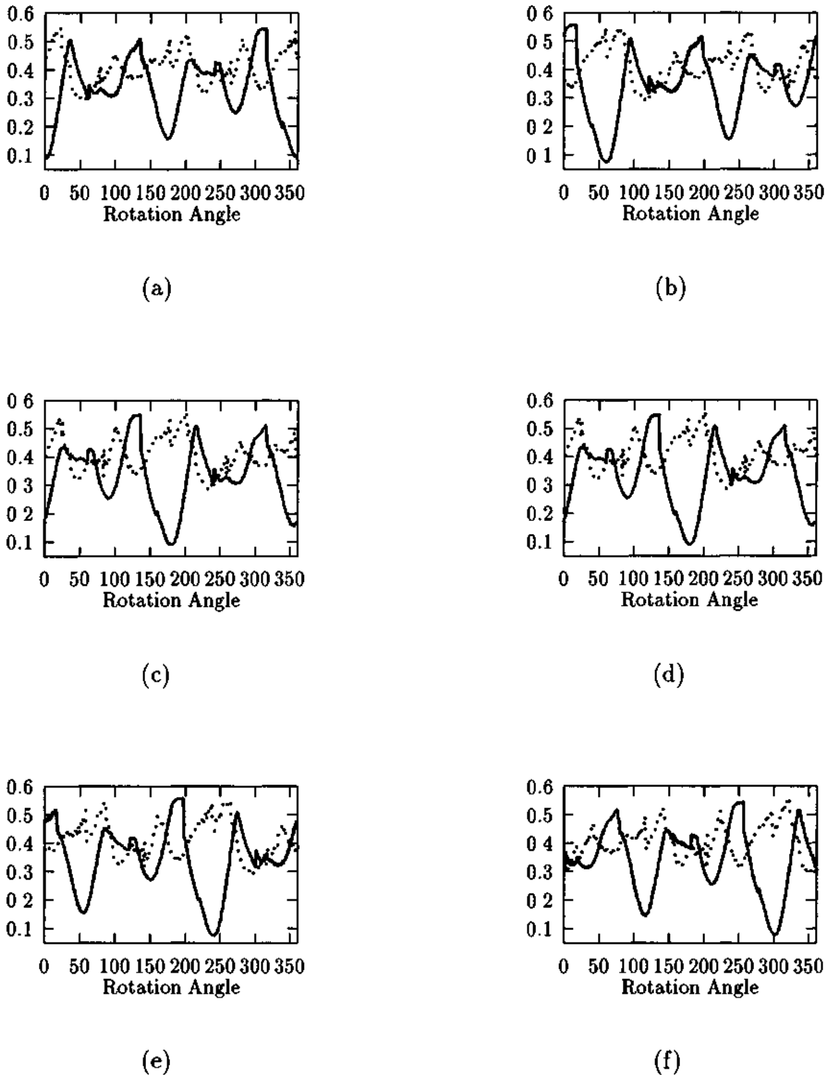


Figure 5.15 $C'^{(1)}_{\beta}$ for characters in Fig. 5.13: (a) the original character, (b) the character is rotated by -60^0, (c) the character is rotated by -120^0, (d) the character is rotated by -180^0, (e) the character is rotated by -240^0, and (f) the character is rotated by -300^0.

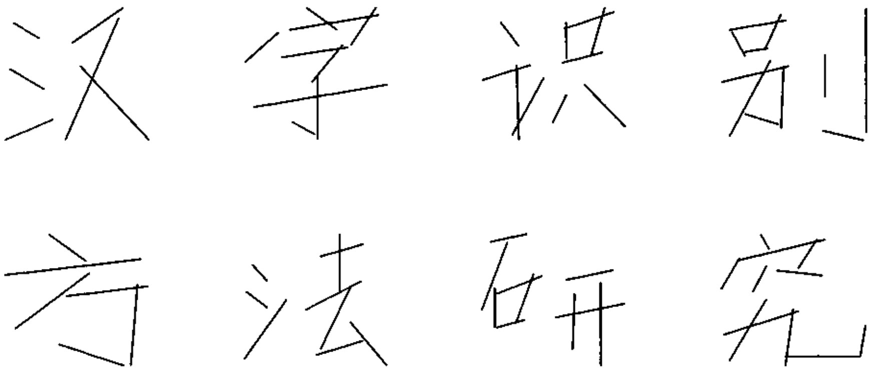


Figure 5.16 Character Prototypes.

5.6 Applications

5.6.1 On-line Handwritten Chinese Character Recognition

In the preceding sections, the line pattern matching algorithms are described for matching two Chinese characters. In this subsection, we show our experimental results on a relatively large character database. We used 8 Chinese characters shown in Fig. 5.16 as prototypes. Note that only one prototype per class is used, so the matching procedure is computationally efficient. We collected 128 samples for testing, with 16 characters in each class. Some testing samples are shown in Fig. 5.17. The characters were written on a tablet with an electronic pen. For each character, we only recorded the positions of two end points of each stroke. That is, the order of drawing the strokes and the direction of drawing a stroke were not used for matching. Using the 8 prototypes, all 128 testing characters are recognized correctly. In theory, the matching algorithms can also be used for off-line character recognition if the character strokes can be thinned and extracted so that a character can be represented as a line pattern.

5.6.2 Point Pattern Matching

A point pattern can be converted to a line pattern by joining selected or all possible pairs of points in the pattern [11]. Figure 5.18 shows three point patterns and their corresponding line patterns. We have joined all pairs of points in each of the three patterns. Note that in Fig. 5.18(b), along the diagonal direction there are three lines, one from the upper right point to the lower left point which overlaps with two

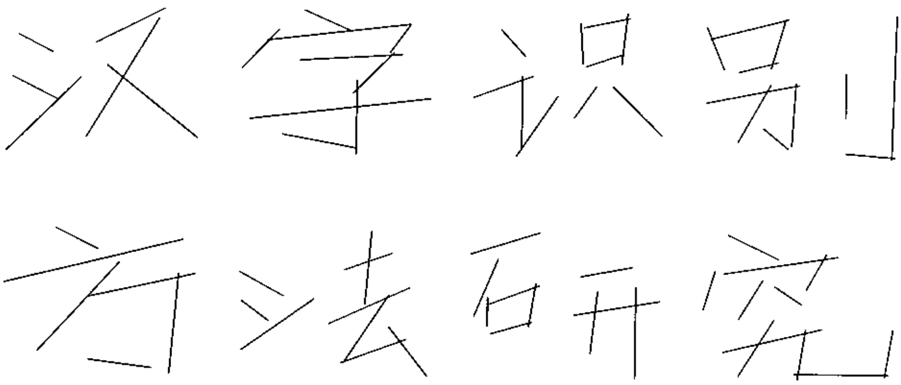


Figure 5.17 Examples of testing samples.

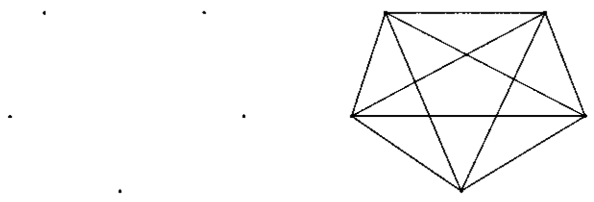
lines from the center point to the upper right and the lower left points respectively. Similarly, three lines along the cross diagonal direction also overlap together. In practical applications, if there are a large number of points in the pattern, we can make use of only a small number of representative lines for matching [11].

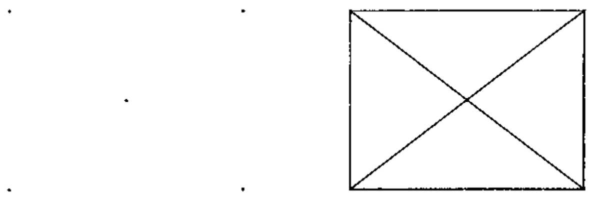
We consider the patterns in Figs. 5.18(a) and 5.18(b) as prototypes, and that in Fig. 5.18(c) as a testing sample which is a distorted version of the pattern in Fig. 5.18(b). In each pattern there are 10 lines for 5 points. The line matching costs are shown in Tables 5.7 and 5.8. Clearly the test pattern is classified correctly although it is distorted.

5.7 Concluding Remarks

Efficient line matching procedures are useful for solving many pattern recognition problems. In this chapter, we have presented a fuzzy logic based method for line matching. In this method, we make use of membership functions to describe position, orientation and relation similarities between different line segments. The position and orientation similarities can be considered as first order measures and the relation similarities can be considered as second order ones. Combining these similarities, we can match a line in the input sample with a line in a prototype if the two lines have similar locations and orientations and similar relations with other lines in the patterns.

Our method can be modified to deal with noisy and rotated patterns. For noisy patterns, we can identify missing and extra line segments by analyzing the matching


(a)


(b)

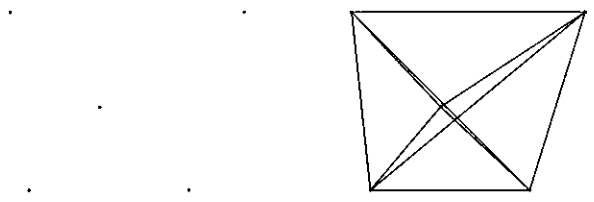

(c)

Figure 5.18 Point patterns and their corresponding line patterns. (a) and (b) are two prototype patterns and (c) is a test pattern.

Table 5.7 Line matching cost matrix $C_{ij}^{(1)}$ for the prototype in Fig. 5.18(a) and the input character in Fig. 5.18(c).

i	$C_{i1}^{(k)}$	$C_{i2}^{(k)}$	$C_{i3}^{(k)}$	$C_{i4}^{(k)}$	$C_{i5}^{(k)}$	$C_{i6}^{(k)}$	$C_{i7}^{(k)}$	$C_{i8}^{(k)}$	$C_{i9}^{(k)}$	$C_{i10}^{(k)}$	j_i	$C_{ij_i}^{(k)}$	$L_{ij_i}^{(k)}$
1	0.76	0.94	1.40	0.75	0.64	1.69	0.63	1.60	0.93	1.87	7	0.63	0.51
2	1.31	1.61	0.69	0.94	1.48	0.88	1.00	2.04	1.90	1.00	3	0.69	0.51
3	1.39	0.55	1.00	0.89	0.36	1.32	0.81	1.58	0.62	1.51	5	0.36	0.29
4	1.65	1.22	0.48	0.99	1.09	0.48	1.05	1.72	1.48	0.80	6	0.48	0.39
5	0.69	1.48	1.00	1.26	0.84	1.47	0.81	1.12	1.37	1.56	1	0.69	0.79
6	1.78	0.38	1.02	0.83	1.02	1.25	1.56	1.53	0.66	1.26	2	0.38	0.27
7	1.84	1.23	0.50	1.03	1.01	1.30	1.56	1.30	1.06	0.54	3	0.50	0.38
8	1.48	1.42	1.03	1.71	0.60	1.23	0.90	0.87	0.82	0.93	5	0.60	0.54
9	1.95	1.30	1.15	1.76	0.99	0.50	1.02	1.43	1.18	0.91	6	0.50	0.34
10	1.33	1.57	1.42	1.83	1.19	1.62	1.57	0.12	1.05	1.16	8	0.12	0.07

Table 5.8 Line matching cost matrix $C_{ij}^{(2)}$ for the prototype in Fig. 5.18(b) and the input character in Fig. 5.18(c).

i	$C_{i1}^{(k)}$	$C_{i2}^{(k)}$	$C_{i3}^{(k)}$	$C_{i4}^{(k)}$	$C_{i5}^{(k)}$	$C_{i6}^{(k)}$	$C_{i7}^{(k)}$	$C_{i8}^{(k)}$	$C_{i9}^{(k)}$	$C_{i10}^{(k)}$	j_i	$C_{ij_i}^{(k)}$	$L_{ij_i}^{(k)}$
1	0.08	1.56	1.46	1.19	1.24	1.61	0.78	1.32	1.62	1.93	1	0.08	0.04
2	1.58	0.05	1.21	0.76	0.87	1.29	1.33	1.63	0.72	1.53	2	0.05	0.03
3	1.65	1.23	0.06	0.79	0.79	0.89	1.19	1.54	1.31	0.53	3	0.06	0.04
4	1.15	0.92	0.65	0.09	1.10	1.33	1.21	1.87	1.45	1.14	4	0.09	0.05
5	1.34	1.14	0.73	1.35	0.18	1.16	0.50	1.25	0.71	1.21	5	0.18	0.15
6	1.87	1.23	1.01	1.47	1.17	0.18	1.08	1.67	1.41	1.07	6	0.18	0.11
7	1.06	1.45	1.17	1.44	0.74	0.97	0.22	1.64	1.22	1.62	7	0.22	0.15
8	1.31	1.56	1.40	1.83	1.17	1.59	1.55	0.03	1.04	1.14	8	0.03	0.02
9	1.54	0.82	1.27	1.45	0.40	1.58	1.05	0.94	0.08	1.39	9	0.08	0.06
10	1.99	1.55	0.44	1.25	1.30	0.69	1.54	1.24	1.48	0.06	10	0.06	0.03

costs of individual lines. For rotated patterns, we can carry out a rough matching first based orientation and relation similarities which can be computed very efficiently. From the rough matching result, we can select several best matching prototypes for finer matching to obtain the final recognition result.

Chapter 6

Fuzzy Rule-based Systems

6.1 Introduction

It is believed that human recognition practice may adopt a set of linguistic rules. The rule-based symbolic processing could be the best platform to explain human intelligence in learning, object recognition, natural language understanding, and a variety of other activities. These rules can be extracted from expert knowledge or learned from examples. A linguistic rule can be described in the form

> IF (a set of conditions are satisfied),
> THEN (a set of consequences can be inferred).

Since the antecedents and the consequents of an IF-THEN rule are usually associated with fuzzy concepts (linguistic terms), they are often in the form of *fuzzy rules.* For a classification related problem, there is one consequence (a class label) inferred from a fuzzy rule in most situations.

In daily life, quite often we use some sort of fuzzy rules to interpret what we perceive to understand the world around us. For example, to describe the flower in a photo of a university campus shown in Fig. 6.1, we probably apply a rule like:

> IF a region is *rather colorful* AND *very attractive* AND *highly patterned* AND the region is *somewhat close to* the regions of other plants,
> THEN it is a *flower.*

Attributes such as "rather colorful", "very attractive", and "highly patterned" defy precise definition, and they are best modeled by fuzzy sets. Similarly, spatial relationships such as "close to" are difficult to model using the traditional all-or-nothing techniques. It is believed that the applications of fuzzy set algorithms to high level vision will produce more realistic solutions.

In a fuzzy rule system for segmenting a map image shown in Fig. 6.2 into fore-

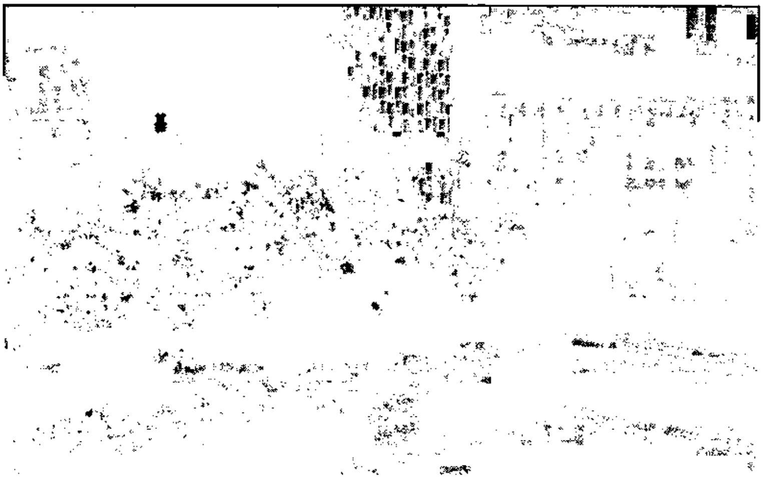


Figure 6.1 A photo of a university campus (originally in color).

ground (characters and lines) and background, a typical rule may look like:

IF a pixel has a *high grey level* AND a *small variance* in a window around the pixel AND *low local contrast*,
THEN it belongs to *background*.

The attributes *high grey level*, *small variance* and *low local contrast* are also better modeled by fuzzy sets.

Membership functions discussed in Chapter 2 are the basis for developing a fuzzy rule-based recognition system. In this chapter, generation and minimization of fuzzy rules, and various defuzzification methods will be discussed. Wang and Mendel have proposed a straightforward algorithm for learning fuzzy rules from numeral data, which will be discussed in Section 6.2. In Section 6.4, we shall present a technique to derive fuzzy rules from the decision trees generated by Quinlan's ID3 algorithm [61]. Generating fuzzy rules from training a neural network has been presented by many researchers [62], [63], [64], [65], [66], [67], [68], [69], [70]. Section 6.4 covers one such approach proposed by Krishnapuram *et al* [71], [72]. Section 6.5 deals with the minimization of fuzzy rules. In particular, a technique based on Karnaugh maps proposed by Hung and Fernandez [73] will be discussed. Section 6.6 covers various techniques for the defuzzification of a fuzzy rule-based recognition system. In particular, a two-layer perceptron based technique to achieve the optimal defuzzification will be discussed. Section 6.7 reports three real-world applications of fuzzy rule recog-

Figure 6.2 A map image.

nition systems, map image segmentation, recognition of printed upper-case English characters, and handwritten digit recognition.

6.2 Learning from Examples

Many methods have been proposed to generate fuzzy rules by learning from numerical data. A method proposed by Wang and Mendel [74] is the most straightforward.

Suppose a set of desired input-output data pairs (the training data) are available:

$$(x_1^{(1)}, x_2^{(1)}; o^{(1)}), (x_1^{(2)}, x_2^{(2)}; o^{(2)}), \ldots \tag{6.1}$$

where $x_1^{(i)}$ and $x_2^{(i)}$ are inputs, and $o^{(i)}$ is the output for the ith data pair. The simple two-input one-output case is chosen for simplicity in the discussion. The method can be easily extended to multi-input and multi-output cases. The task here is to generate a set of fuzzy rules from the training data of (6.1), and use these fuzzy rules to determine a mapping $o = f(x_1, x_2)$. The method is originally proposed for deriving fuzzy rules for function approximation. We have extended it for solving the problems of classification and recognition [24], [26], [75]. The method consists of the following five steps:

Step 1: Fuzzify the Input Space

Find the domain intervals of x_1 and x_2, $[x_1^{(l)}, x_1^{(h)}]$ and $[x_2^{(l)}, x_2^{(h)}]$ and partition each input domain interval into N_i regions ($i = 1, 2$). The regions are labeled as

R_1 (the lowest), R_2, ..., R_{N_i} (the highest).

A membership function is adopted for each fuzzy region. These membership functions can take the commonly used triangular and trapezoidal shapes or be generated by using the techniques as described in Chapter 2. For a classification problem, a hard partition instead of a fuzzy one is adopted for the output. In the case of L-class problem, the outputs are $\{C_1, C_2, ..., C_L\}$, which is a crisp set.

Step 2: Generate Fuzzy Rules from Given Data Pairs

1. Determine the membership grades of given $x_1^{(i)}$ and $x_2^{(i)}$ in different input fuzzy subsets (linguistic labels). For example, in Fig. 6.3, $x_1^{(1)}$ has a grade of 0.75 in R_2, 0.25 in R_3, and zero in all other regions. Similarly, $x_2^{(2)}$ in Fig. 6.3 has a grade of 1 in R_3 and zero in all other regions.

2. Assign given inputs to the regions with the maximum membership grade. In our example, $x_1^{(1)}$ is considered being R_2, and $x_2^{(2)}$ is considered being R_3.

3. Produce a rule from each input-output data pair, for example,

   $(x_1^{(1)}, x_2^{(1)}; o^{(1)}) \Rightarrow$
   $x_1^{(1)}(0.75 \text{ in } R_2 \text{ (max)}), x_2^{(1)}(0.6 \text{ in } R_5 \text{ (max)}); o^{(1)}(1.0 \text{ in } C_2 \text{ (max)}) \Rightarrow$
   Rule 1: IF x_1 is R_2 AND x_2 is R_5, THEN o is C_2.

   $(x_1^{(2)}, x_2^{(2)}; o^{(2)}) \Rightarrow$
   $x_1^{(2)}(0.87 \text{ in } R_4 \text{ (max)}), x_2^{(2)}(1.0 \text{ in } R_3 \text{ (max)}); o^{(2)}(1.0 \text{ in } C_1 \text{ (max)}) \Rightarrow$
   Rule 2: IF x_1 is R_4 AND x_2 is R_3, THEN o is C_1.

Step 3: Assign a Degree to Each Rule

As mentioned above, each data pair generates one rule. Usually there are a large number of data pairs available, so it is very likely that some conflicting rules are produced. The conflicting rules have the same IF part but different THEN parts. One way to solve this problem is to assign a *soundness degree* to each rule generated and accept only the rule with the greatest soundness degree from a group of conflicting rules. Suppose that the ith rule is "IF x_1 is R_{i1} and x_2 is R_{i2}, THEN o is C_i" and the soundness degree of the rule is denoted by $D(\text{Rule}_i)$. Two strategies can be followed to assign a soundness degree to a rule.

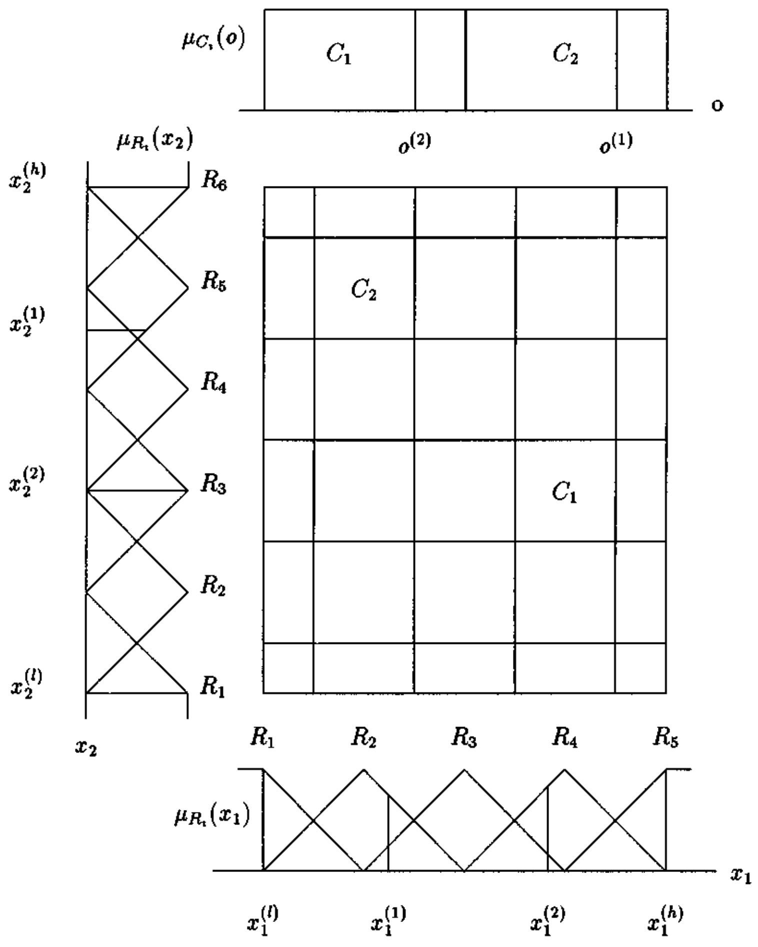


Figure 6.3 Fuzzy partitions and the rule bank for a two-input two-category classification problem.

Strategy 1: The degree is determined by the membership grades of inputs and output(s):

$$D(\text{Rule}_i) = \mu_{R_{i1}}(x_1)\mu_{R_{i2}}(x_2)\mu_{C_{i1}}(o). \tag{6.2}$$

As the crisp sets are adopted for the output space, we always have $\mu_{C_{i1}}(o) = 1$. Therefore,

$$D(\text{Rule}_i) = \mu_{R_{i1}}(x_1)\mu_{R_{i2}}(x_2). \tag{6.3}$$

For example, Rule 1 has the degree

$$\begin{aligned} D(\text{Rule}_1) &= \mu_{R_2}(x_1)\mu_{R_5}(x_2) \\ &= 0.75 \times 0.6 = 0.45 \end{aligned} \tag{6.4}$$

If we have *a priori* knowledge about the data pairs, we can assign a degree of belief to the usefulness of each data pair. Suppose that the data pair $((x_1^{(1)}, x_2^{(1)}; o^{(1)})$ has a degree $m^{(1)}$, then we have

$$D(\text{Rule}_1) = \mu_{R_2}(x_1)\mu_{R_5}(x_2)m^{(1)}. \tag{6.5}$$

This strategy will be used if there are only a small number of training samples available.

Strategy 2: The degree is determined by the ratio of the number of data pairs which support the rule (N_{Rule_i}) and the total number of patterns which have the same IF part (N_{If_i}).

$$D(\text{Rule}_i) = \frac{N_{\text{Rule}_i}}{N_{\text{If}_i}}. \tag{6.6}$$

This strategy works better when a large number of training samples are available. Actually, by using this frequency based degree, we incorporate the statistical information into fuzzy systems resulting in more reliable decision making.

Step 4: Create a Combined Rule Bank

The form of the rule bank is shown in Fig. 6.3. The boxes of the bank are filled with fuzzy rules from either those generated from numerical data or linguistic rules extracted based on expert knowledge (a linguistic rule also has a degree of soundness which is assigned by an expert). Rules 1 and 2 shown in the example can be filled in the bank at boxes (R_2, R_5) and (R_4, R_3) with output labels C_2 and C_1, respectively. If a linguistic rule is an "and" rule, it fills only one box of the rule bank; but if a linguistic rule is an "or" rule, it fills all the boxes in the rows or columns corresponding to the regions of the IF part.

Step 5: Determine the Mapping by Using a Defuzzification Method

A defuzzification method is adopted to determine the mapping between inputs and the output because of two reasons:

1. Not all regions would be filled by fuzzy rules. A defuzzification scheme has to be used to determine the output of the input non-fuzzy regions based on neighboring rules; and

2. Using a single rule may not be reliable in decision making.

Note that various minimization techniques can be applied to reduce the number of fuzzy rules so that the system complexity is reduced. The optimized defuzzification should be used to keep the classification performance from degradation due to the minimization of fuzzy rules.

6.3 Decision Tree Approach

As a machine learning technique, the decision tree approach, has been widely used in pattern recognition because of its ease of implementation and its comprehensibility.

6.3.1 Decision Trees and Rules

This section introduces an algorithm to generate decision trees by learning from training data. The following discussion is mainly based on the work by Quinlan [61].

Assume each training pattern contains a set of input features (F_i, $i = 1, 2, ..., n$) and an associated classification (C_k). Figure 6.4 shows a decision tree produced for a four-input problem (six decision rules). Each node represents a feature F_i while each branch, f_{ij}, represents the jth region of values of feature F_i. For a feature with discrete values, f_{ij} stands for the jth value of feature F_i. For a feature with continuous values, f_{ij} stands for one of four regions of values: $F_i < f_{ij}$, $F_i \leq f_{ij}$, $F_i \geq f_{ij}$, and $F_i > f_{ij}$. A leaf node, L_j, contains a classification $C(L_j)$. There are all together six rules represented by the decision tree shown in Fig. 6.4. One of these rules (the highlighted path) is:

IF $F_1(f_{11})$ is true AND $F_2(f_{22})$ is true AND $F_4(f_{42})$ is true,
THEN the class is $C(L_6)$.

$F_i(f_{ij})$ is defined as

$$F_i(f_{ij}) = \begin{cases} 1 \text{ (true)} & \text{if } F_i \text{ is in the region } f_{ij} \text{ of feature } F_i, \\ 0 \text{ (false)} & \text{otherwise.} \end{cases} \tag{6.7}$$

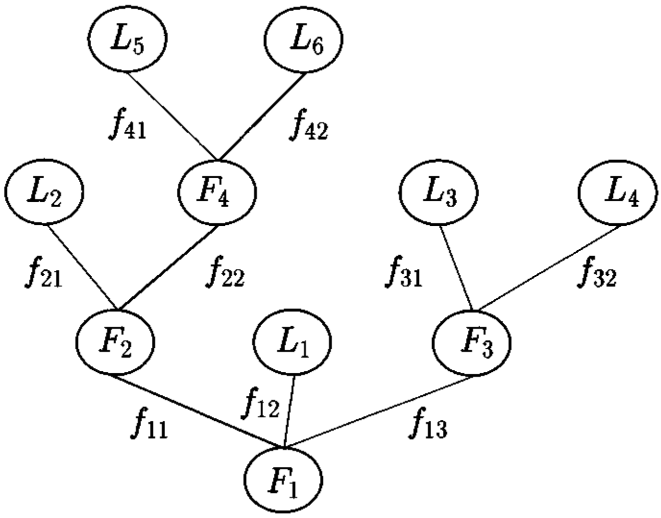


Figure 6.4 A decision tree.

Let M be an L-class problem which contains m_i objects from class C_i ($i = 1, ..., L$). In the learning procedure, the decision tree generation technique adopts an information-based method that depends on the following two assumptions [61]:

1. Any correct decision tree for M will classify objects in the same proportion as their representation in M. An arbitrary object will be determined to belong to class C_i with probability $(\frac{m_i}{\sum_{l=1}^{L} m_l})$.

2. When a decision tree is used to classify an object, it returns a class. A decision can thus be regarded as a source of a message 'C_1', 'C_2', ..., or 'C_L', with the expected information needed to generated this message given by

$$I(M) = -\sum_{i=1}^{L}(\frac{m_i}{\sum_{l=1}^{L} m_l})\log_2(\frac{m_i}{\sum_{l=1}^{L} m_l}). \tag{6.8}$$

If feature F_i with values $\{f_{i1}, f_{i2}, ..., f_{iv}\}$ is used for the root of the decision tree, it will partition M into $\{M_{i1}, M_{i2}, ..., M_{iv}\}$ where M_{ij} contains m_{ij} objects in M that have value f_{ij} of F_i. The expected information required for the subtree for M_{ij} is $I(M_{ij})$. The expected information required for the tree with F_i as root is then obtained as the weighted average,

$$E(F_i) = \sum_{j=1}^{v} \frac{m_{ij}}{\sum_{l=1}^{v} m_{il}} I(M_{ij}) \tag{6.9}$$

where the weight for the jth branch is the proportion of the objects in M that belong to M_{ij}. The information gained by branching on F_i is therefore,

$$\text{gain}(F_i) = I(M) - E(F_i). \tag{6.10}$$

A good rule of thumb would be to choose that feature to branch on which gains the most information. Since $I(M)$ is constant for all features, maximizing the gain is equivalent to minimizing $E(F_i)$. The technique, therefore, examines all candidate features and choose F_{i*} to minimize $E(F_i)$, forms the tree, and then uses the same process recursively to form decision trees for the residual subsets M_{i*1}, M_{i*2}, ..., M_{i*v}.

Now let us look at an example in which decisions have to be made whether a pair of contact lenses are used and if so which type of contact lenses is adopted based on four features: patient's age, spectacle prescription, astigmatic, and tear production rate. The three outcomes are: the patient should be fitted with (1) hard contact lenses, (2) soft contact lenses, or (3) none contact lenses. The database is from a highly simplified problem donated by Benoit Julien and is available at the time of writing, in the Machine Learning Databases at the internet site "*ics.uci.edu*". Table 6.1 lists the training set.

We consider the Age feature with a set of values A = {young, pre-presbyopic, presbyopic}. Eight of the 24 objects in M have the value of *young*, that is, $m_{11} = 8$. Among them, two each are from classes *hard* and *soft*, and four from class *none*, so

$$I(M_{11}) = -\frac{2}{8}\log_2\frac{2}{8} - \frac{2}{8}\log_2\frac{2}{8} - \frac{4}{8}\log_2\frac{4}{8} = 1.500 \tag{6.11}$$

and similarly, $I(M_{12}) = 1.298$ and $I(M_{13}) = 1.812$. Then we have

$$E(F_1) = \sum_{j=1}^{3} \frac{m_{1j}}{\sum_{l=1}^{3} m_{1l}} I(M_{1j}) = \frac{8}{24} \times 1.500 + \frac{8}{24} \times 1.298 + \frac{8}{24} \times 1.812 = 1.537 \tag{6.12}$$

Similar analysis gives $E(F_2) = 1.287$, $E(F_3) = 0.950$ and $E(F_4) = 0.777$. Because of $F_4 = \min_i E(F_i)$, F_4 (tear-production rate) will be chosen as the feature for the root of the decision tree. It has two branches: *reduced* and *normal*. For the branch of *reduced*, all patterns have the class of *none* so a leaf is obtained. On the other hand, sub-branches are required for the branch of *normal*. This process continues until all branches contain patterns from the same class or the features have been used up.

Table 6.1 Training samples for generating decision rules for deciding whether a pair of contact lenses should be used (TRR: tear-production rate).

No.	Features				Class
	Age	Spectacle	Astigmatic	TRR	
1	young	myope	no	reduced	none
2	young	myope	no	normal	soft
3	young	myope	yes	reduced	none
4	young	myope	yes	normal	hard
5	young	hypermetrope	no	reduced	none
6	young	hypermetrope	no	normal	soft
7	young	hypermetrope	yes	reduced	none
8	young	hypermetrope	yes	normal	hard
9	pre-presbyopic	myope	no	reduced	none
10	pre-presbyopic	myope	no	normal	soft
11	pre-presbyopic	myope	yes	reduced	none
12	pre-presbyopic	myope	yes	normal	hard
13	pre-presbyopic	hypermetrope	no	reduced	none
14	pre-presbyopic	hypermetrope	no	normal	soft
15	pre-presbyopic	hypermetrope	yes	reduced	none
16	pre-presbyopic	hypermetrope	yes	normal	none
17	presbyopic	myope	no	reduced	none
18	presbyopic	myope	no	normal	none
19	presbyopic	myope	yes	reduced	none
20	presbyopic	myope	yes	normal	hard
21	presbyopic	hypermetrope	no	reduced	none
22	presbyopic	hypermetrope	no	normal	soft
23	presbyopic	hypermetrope	yes	reduced	none
24	presbyopic	hypermetrope	yes	normal	none

The generated decision tree is:

```
tear-production-rate = reduced: none
tear-production-rate = normal:
                        astigmatic = no: soft
                        astigmatic = yes:
                                        spectacle = myope: hard
                                        spectacle = hypermetrope:
                                                        age = young: hard
                                                        age = pre-presbyopic: none
                                                        age = presbyopic: none
```

There are altogether six rules which can extracted from the decision tree. They are:

Rule 1:	IF	tear-production rate is *reduced*,	THEN	the prescription is *none* contact lenses.
Rule 2:	IF	tear-production rate is *normal* AND astigmatic is *no*,	THEN	the prescription is *soft* contact lenses.
Rule 3:	IF	tear-production rate is *normal* AND astigmatic is *yes* AND spectacle is *myope*,	THEN	the prescription is *hard* contact lenses.
Rule 4:	IF	tear-production rate is *normal* AND astigmatic is *yes* AND spectacle is *hypermetrope* AND age is *young*,	THEN	the prescription is *hard* contact lenses.
Rule 5:	IF	tear-production rate is *normal* AND astigmatic is *yes* AND spectacle is *hypermetrope* AND age is *pre-presbyopic*,	THEN	the prescription is *none* contact lenses.
Rule 6:	IF	tear-production rate is *normal* AND astigmatic is *yes* AND spectacle is *hypermetrope* AND age is *presbyopic*,	THEN	the prescription is *none* contact lenses.

The decision tree classifies 23 patterns correctly and one incorrectly. Pattern 18 should have the output of *none* but the decision tree gives the output of *soft*.

Compared with fuzzy rules generated by learning from examples discussed in Section 6.2, the rules produced by the decision tree approach are more meaningful and flexible in the sense that various rule sizes are adopted.

A rule extracted from the decision tree approach for numeral classification (mixed discrete, symbolic and continuous features) looks like (the reader is referred to Section 6.7.3 for more detailed discussion):

IF the type for the longest segment is *circle* AND the type for second longest segment is *C curve* AND the normalized y coordinate of the image center > 0.586,
THEN it is digit 6.

6.3.2 Simplified Decision Trees

Quinlan proposed a two-stage simplification scheme when an ID3 tree is converted to a set of production rules [76]. Obviously, the path from the root to each leaf of a decision tree corresponds to a production rule:

IF $F_{i_1}(f_{i_1 j_1})$ is true AND $F_{i_2}(f_{i_2 j_2})$ is true AND ... AND $F_{i_n}(f_{i_n j_n})$ is true,
THEN the class is C_k.

The first stage examines each production rule and generalizes it by dropping conditions from its IF part based on a significance level test. At each time, the condition which has the least relevance to classification is discarded if it fails the significant level test. The process is repeated until no more conditions can be dropped. After the first stage process, some leaves produce identical rules while other leaves generate vacuous rules with all conditions dropped. The number of rules is generally smaller than the number of leaves.

The second stage evaluates how well the rules will function as a set. For each rule in turn, we now determine how the remaining rules classify the training set if this rule is deleted. Those rules whose omission would not increase the misclassification rate or would even reduce it are least useful. These rules are discarded and the process is repeated. After the second stage process, the number of rules decreases further.

6.3.3 Fuzzified Decision Rules

Decision rules work well when the input data are accurate. However, their performance degrades when input data are uncertain or noisy. In these cases, fuzzy trees or rules can be used to improve the classification performance [77], [78].

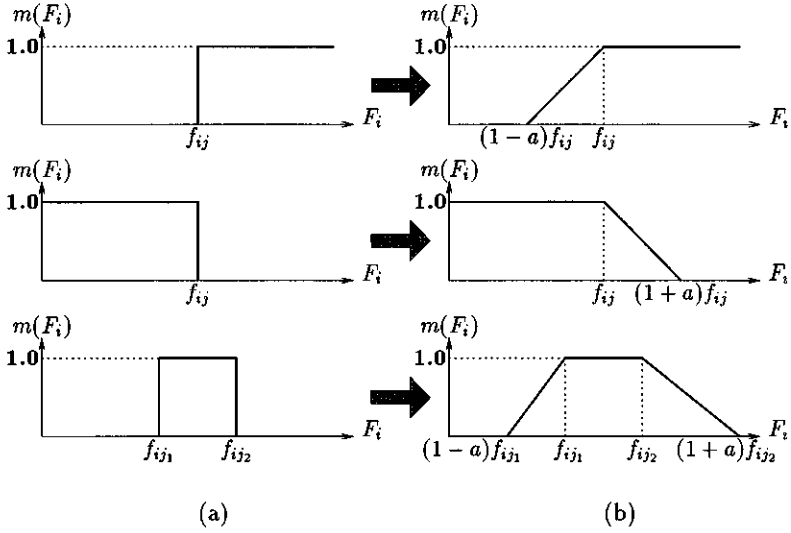


Figure 6.5 (a) Feature regions and (b) their membership functions.

For a decision rule, antecedent conditions are either TRUE (1) or FALSE (0), and only one rule is chosen to perform classification. To fuzzify these rules, we have to choose or derive membership functions from the training date and use the membership grades instead of a binary value, 0 or 1 to measure the matching degree. Defuzzification is then used to integrate these fuzzy rules (assume that all rules contribute to the classification).

For a feature with L discrete values (symbols), an $L \times L$ fuzzy grade matrix M has to be constructed. On the other hand, membership functions have to be used for a feature with continuous values. As mentioned in Section 6.3.1, there are four basic types of value regions used for a feature with continuous values in decision rules. They are $F_i > f_{ij}$, $F_i \geq f_{ij}$, $F_i < f_{ij}$, and $F_i \leq f_{ij}$ (see Fig. 6.5(a)). The third case in Fig. 6.5 is actually the combination of cases 1 and 2. For $F_i > f_{ij}$ and $F_i \geq f_{ij}$, we define the membership function as

$$m(F_i) = \begin{cases} 0.0 & \text{if } F_i \leq (1-a)f_{ij}, \\ \frac{F_i-(1-a)f_{ij}}{af_{ij}} & \text{if } (1-a)f_{ij} < F_i \leq f_{ij}, \\ 1.0 & \text{if } F_i > f_{ij} \end{cases} \tag{6.13}$$

where a is an extension factor. For $F_i < f_{ij}$ and $F_i \leq f_{ij}$, we have

$$m(F_i) = \begin{cases} 1.0 & \text{if } F_i \leq f_{ij}, \\ \frac{(1+a)f_{ij}-F_i}{af_{ij}} & \text{if } f_{ij} < F_i \leq (1+a)f_{ij}, \\ 0.0 & \text{if } F_i > (1+a)f_{ij}. \end{cases} \tag{6.14}$$

The resulting membership functions are shown in Fig. 6.5(b). By using the membership values instead of binaries 0 and 1, the decision rules become a set of fuzzy rules which can be more tolerant to noise and other distortions and thus have higher classification power. A defuzzification method to be discussed later can be applied to perform the classification using these soft decision rules.

6.4 Fuzzy Aggregation Network Approach

The ability of neural networks to generalize a problem from existing training data by using a learning algorithm is the most valuable property. Recently, there has been considerable interest in combining fuzzy logic and neural network techniques. There are typically four ways to use neural networks in developing a fuzzy system: (1) using neural networks to learn or tune membership functions as discussed in Section 2.4.2; (2) using neural networks to determine the rules themselves, such as extracting rules from a trained feedforward neural network [79]; (3) developing special node combination schemes based on fuzzy set connectives; and (4) utilizing neural networks to perform a fuzzy logic inference directly.

Several different aggregation operators can be used to integrate membership values. Based on their aggregation behavior, these connectives can be grouped into three classes: union connectives, intersection connectives, and compensative connectives. The generalized mean operator given below can achieve different connectives by varying the value of parameter p [71]:

$$g(x_1, x_2, ..., x_n; p, w_1, w_2, ..., w_n) = \left(\sum_{i=1}^{n} w_i x_i^p\right)^{1/p}. \tag{6.15}$$

The w_i's are the relative importance factors for different inputs where

$$w_1 + w_2 + ... + w_n = 1. \tag{6.16}$$

The generalized mean has several attractive properties. For example, the mean value always increases with an increase in p. Thus, by varying the value of p between $-\infty$ and $+\infty$, we can obtain all types of aggregation. In the extreme cases, this operator can be used as intersection (*min*) or union (*max*). Also, it can be shown that $p = -1$ gives the harmonic mean and $p = 1$ gives the arithmetic mean.

After suitable membership functions are chosen or derived from the data, a three layer fuzzy aggregation network, which was proposed by Krishnapuram *et al* [72], can be used to produce a compact set of rules with conjunctive and disjunctive antecedent clauses. In this network, the input layer consists of K (the number of inputs) groups of nodes with the kth group consisting of L_k nodes (equal to the number of linguistic terms for the kth input). The ith node in the kth group corresponds to the ith linguistic label in the domain of x_k and uses h_{ki} (the membership function of the ith linguistic label of input k) as the activation function. The hidden layer has K groups of L nodes (L is the number of outputs which is equal to the number of classes). The ith node in group k in the input layer is connected to the jth node in the corresponding group in the hidden layer if h_{ki} has a non-empty intersection with $m_k^j(x_k)$ (the membership functions of the jth class over input k). In practice, a threshold is used to decide whether it is a non-empty intersection. The jth node of each group in the hidden layer is connected to the jth node of the output layer for $j = 1, ..., L$. Each node in the hidden and output layers uses a fuzzy aggregation function (such as the generalized mean) as the activation function. Figure 6.6 shows an artificial classification problem with two inputs and two classes ("black square" and "black circle"). Figure 6.7 shows an aggregation network for solving this problem. Each input has three linguistic labels denoted by L, M and H and adopts a set of evenly distributed triangular membership functions. Because there is no overlap between the membership function for the linguistic label "L" and the class 1 membership function over the input X_2, there is no connection between the input layer node "L" of input X_2 to node 5 in the hidden layer. The initial (approximate) network includes all solid and dash connections. After the training, the dash connections are removed because of small weights.

If we interpret a node with the p value of greater than 0 as a union aggregation operator and a node with the p value of smaller than 0 as an intersection aggregation operator, the following rules could possibly be produced after those connections with small weights have been cut off:

Rule 1: IF Input X_1 is (*L* OR *M*) AND Input X_2 is *H*,
THEN the class is *Black Square*.
Rule 2: IF Input X_1 is *H* OR Input X_2 is (*L* OR *M*),
THEN the class is *Black Circle*.

6.5 Minimization of Fuzzy Rules

For pattern recognition problems, a large number of training patterns are often available. Therefore, a large number of fuzzy rules are usually produced when the scheme of learning rules from examples is used. However, we often need to reduce the number of fuzzy rules to make a recognition system more practical and to speed up the processing. There are several different methods proposed in literature to minimize fuzzy rules. These techniques can be put into two categories: minimization before producing rules and minimization after producing rules. Reducing the number of training patterns by using a data clustering technique, reducing the number of features by using a feature selection technique, and reducing the feature fuzzy partitions by generating the membership functions based on the actual data distribution, are some

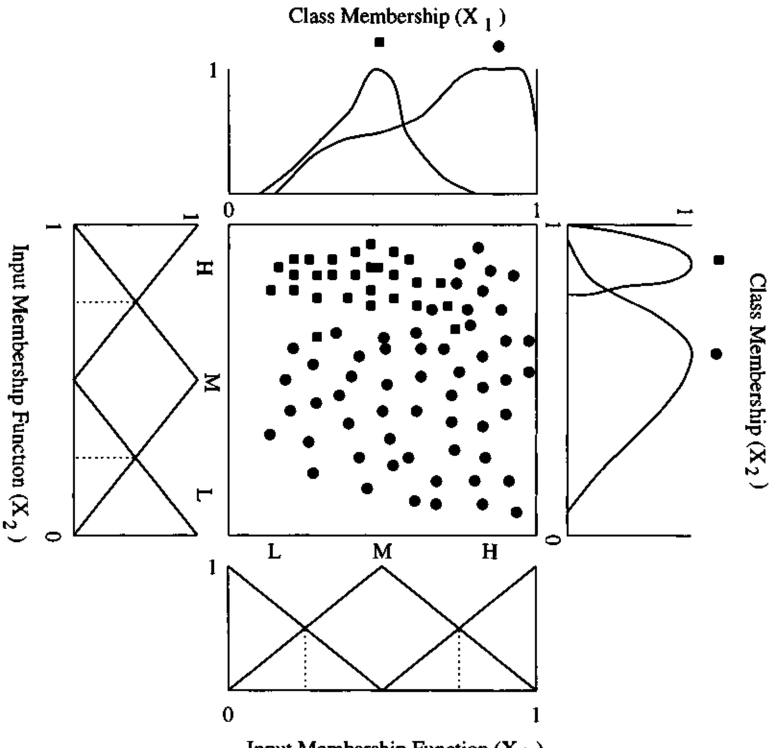


Figure 6.6 Data distribution and corresponding class/input membership functions for a synthesized two-input and two-class (black square and black circle) problem.

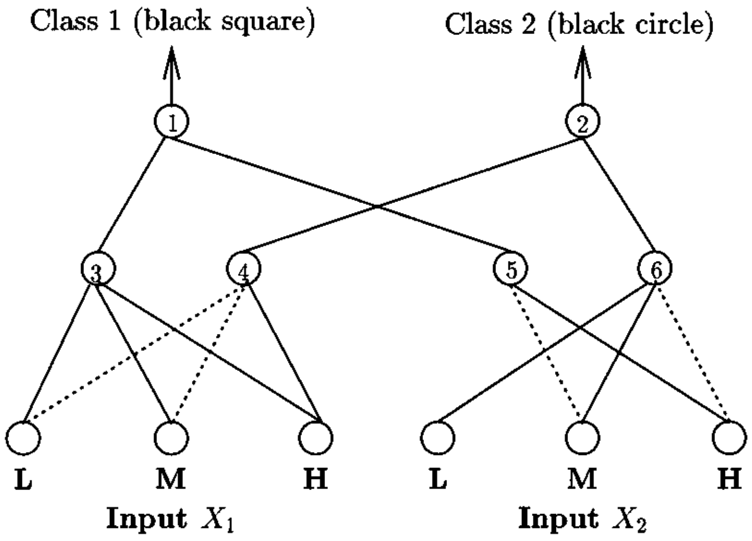


Figure 6.7 Approximate network structure for generating rules and the reduced network after training (dotted line connections will be removed).

methods that can be used to reduce the number of fuzzy rules to be produced. On the other hand, three methods of rule pruning, rule elimination and rule combination can be used to minimize fuzzy rules produced. In one technique, we can eliminate those rules which are supported by a small number of training patterns or which have a low confidential degree. Karnaugh maps, which provide systematic methods for simplifying switching functions in the logic design of digital systems, can also be used to minimize fuzzy rules.

6.5.1 Minimization Based on Data Clustering

There are many techniques available for data clustering. c-means, fuzzy c-means, Kohonen self-organizing maps, and adaptive vector quantization (AVQ) are some of these techniques. We have already discussed in Chapter 2 how to use clustering algorithms to generate the membership functions that reflect the actual data distribution. By using labeled cluster centers instead of all training examples, we can easily reduce the number of fuzzy rules substantially. However, the system performance usually degrades due to the reduced number of fuzzy rules. Therefore, an optimized defuzzification procedure has to be adopted to maintain the system performance.

6.5.2 Minimization Based on Karnaugh Maps

Hung and Fernandez proposed an approach based on Karnaugh Maps for the minimization of fuzzy rules [73]. Before we can make use of the algebraic simplification of switching functions, we need to transform the linguistic descriptors of fuzzy rules into algebraic expressions. Fuzzy rules have fuzzy propositions for their premises and conclusions. Therefore, we have to first convert the linguistic values of fuzzy subsets into binary representation and then display them on a Karnaugh Map. Similar to switching functions, a set of fuzzy rules can also be represented as a minterm expression (standard sum of products), a maxterm expression (standard product of sums), or as an algebraic form. Using the postulates and theorems of Boolean algebra, the minimum sum-of-products or minimum product-of-sums rules can be derived from the originally complicated expressions. To make use of a Karnaugh map for fuzzy rule simplification, we need to represent a set of fuzzy rules as a minterm expression.

The procedures for reducing the number of rules using the Karnaugh map are:

1. The linguistic values (fuzzy subsets) of input and output variables are regarded as crisp values (0 or 1) without considering the overlap between fuzzy subsets.

2. Discretize the input and output variables into n-bit integers, which have 2^n possible values (each linguistic value can be represented as one binary value). Note that some binary values might not be used.

3. We can then minimize the multi-variable functions using the Karnaugh map. For example, the minimum sum-of-products expressions can be derived by using the formula:

$$X_iY_1 + X_iY_2 + \ldots + X_iY_n = X_i. \tag{6.17}$$

where X_i is one of linguistic values of variable X. $Y_1, Y_2, \ldots, Y_n$ are all n linguistic values that Y variable takes. We will show the simplification process based on Karnaugh maps later with two examples.

4. An optimization scheme is usually taken to keep the system performance from degrading due to the reduced number of fuzzy rules.

Now let us see how we can minimize a set of fuzzy rules based on Karnaugh maps for two cases, completely-specified rules and incompletely-specified rules.

(a) Completely-specified Rules:

Suppose that we have two input variables (x_1 and x_2) and one output variable (y). Each of them has four linguistic labels, and for x_1, x_2 and y, the labels are $\{A_1, A_2, A_3, A_4\}$, $\{B_1, B_2, B_3, B_4\}$, and $\{C_1, C_2, C_3, C_4\}$, respectively. The rules are listed below:

IF x_1 is A_1 AND x_2 is B_3, THEN y is C_2.
IF x_1 is A_2 AND x_2 is B_3, THEN y is C_2.
IF x_1 is A_3 AND x_2 is B_1, THEN y is C_3.
IF x_1 is A_3 AND x_2 is B_2, THEN y is C_3.
IF x_1 is A_3 AND x_2 is B_3, THEN y is C_3.
IF x_1 is A_3 AND x_2 is B_4, THEN y is C_3.
IF x_1 is A_4 AND x_2 is B_3, THEN y is C_2.

After discretizing the input and output variables, we can construct a Karnaugh map shown in Fig. 6.8 to represent these rules. The corresponding *minterm* expression is given below:

$$y = A_1B_3 + A_2B_3 + A_3B_1 + A_3B_2 + A_3B_3 + A_3B_4 + A_4B_3. \tag{6.18}$$

From the map, we can see that all minterms in column A_3 has the same output 10 (C_3), so we can use Eq. (6.17) to simplify them. As a result, $(A_3B_1 + A_3B_2 + A_3B_3 + A_3B_4)$ is reduced to A_3. Hence we have

$$y = A_1B_3 + A_2B_3 + A_3 + A_4B_3. \tag{6.19}$$

A set of seven fuzzy rules is now reduced to the following four rules:

		x_1			
		A_1	A_2	A_3	A_4
		00	01	10	11
	B_1 00			10	
	B_2 01			10	
x_2	B_3 10	01	01	10	01
	B_4 11			10	

Figure 6.8 The Karnaugh map representation for a set of completely-specified rules.

IF x_1 is A_1 AND x_2 is B_3, THEN y is C_2.
IF x_1 is A_2 AND x_2 is B_3, THEN y is C_2.
IF x_1 is A_3, THEN y is C_3;
IF x_1 is A_4, AND x_2 is B_3, THEN y is C_2.

(b) Incompletely-specified Rules:

In this case, a set of rules available are the same as the previous example except that the following rule is excluded:

IF x_1 is A_3 AND x_2 is B_3, THEN y is C_3.

In the Karnaugh map shown in Fig. 6.9, we use 'xx' to represent this rule as a "don't care" case.

Similarly, we have a minterm expression given by

$$y = A_1B_3 + A_2B_3 + A_3B_1 + A_3B_2 + A_3B_3(xx) + A_3B_4 + A_4B_3. \tag{6.20}$$

By examining the map, we find that if we consider A_3B_3 having an output of C_3, all minterms in column A_3 have the same output C_3 and they can be reduced to A_3. Also if we consider A_3B_3 having an output of C_2, all minterms in row B_3 have the same output C_2 and they can be reduced to B_3. Because A_3B_3 is a don't care case, we can use it twice to simplify both column A_3 and row B_3. As a result, we have

				x_1		
			A_1	A_2	A_3	A_4
			00	01	10	11
	B_1	00			10	
	B_2	01			10	
x_2	B_3	10	01	01	xx	01
	B_4	11			10	

Figure 6.9 The Karnaugh map representation for a set of incompletely-specified rules.

$$y = A_3 + B_3. \tag{6.21}$$

A set of six rules is now reduced to the following two rules:

IF x_1 is A_3, THEN y is C_3.
IF x_2 is B_3, THEN y is C_2.

The above examples have demonstrated the minimization of two simple problems. However, the technique can be easily extended to more complicated problems.

6.6 Defuzzification and Optimization

Figure 6.10 shows the simplified block diagram of a fuzzy classification system [56]. Suppose we have a set of alternatives $C = \{C_1, C_2, ..., C_M\}$ (classes). A fuzzy subset F over C indicating the degree to which each alternative satisfies our decision criteria and goals. The defuzzification problem defines the strategy of using the fuzzy subsets F to guide us in the selection of one representative element (class) of the set C.

Several defuzzification methods, including maximum matching, maximum accumulated matching, the centroid defuzzification, and a two-layer neural network approach to achieve the optimized defuzzification, are discussed in the following sections.

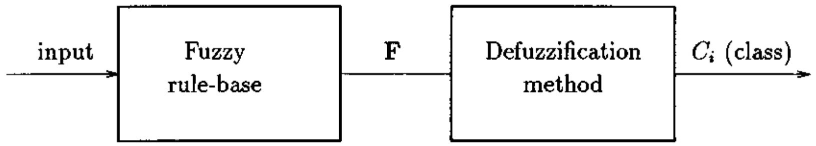


Figure 6.10 The simplified block diagram of a fuzzy rule-based classification system.

6.6.1 Maximum Matching

Suppose that there are K fuzzy rules and among them, K_j rules ($j = 1, 2, ..., L$ and L is the number of classes) produce class C_j. Let D_p^i be the measurement of how the pth pattern matches the antecedent conditions (IF-part) of the ith rule, which is given by the product of membership grades of the pattern in the regions which the ith rule occupies (matching degree), that is,

$$D_p^i = \prod_{l=1}^{n} m_{li} \tag{6.22}$$

where n is the number of inputs and m_{li} is the membership grade of feature l in the fuzzy regions that the ith rule occupies. Let $D_p^m(C_j)$ be the maximum matching degree of the rules (rules $j_l, l = 1, 2, ..., K_j$) generating class C_j, that is,

$$D_p^m(C_j) = \max_{l=1}^{K_j} D_p^{j_l}, \tag{6.23}$$

then the system will output class C_{j_*} provided that

$$D_p^m(C_{j_*}) = \max_{j} D_p^m(C_j). \tag{6.24}$$

If there are two or more classes which achieve the maximum matching degree, we will select the class which has the largest number of fired fuzzy rules (a fired rule has a matching degree of greater than zero).

6.6.2 Maximum Accumulated Matching

For each unlabeled pattern, we accumulate the matching degrees of rules which output the same class. The accumulated matching degree for Class j, $D_p^a(C_j)$, is defined as

$$D_p^a(C_j) = \sum_{l=1}^{K_j} D_p^{j_l}. \tag{6.25}$$

The final decision is to assign the pattern to the class which has the maximum accumulated matching degree. If

$$D_p^a(C_{j_*}) = \max_j D_p^a(C_j) \tag{6.26}$$

then the system will output class j_*.

6.6.3 Centroid Defuzzification

The centroid defuzzification formula to determine the output (O_p) for each input pattern is

$$O_p = \frac{\sum_{i=1}^{K} D_p^i O^i}{\sum_{i=1}^{K} D_p^i} \tag{6.27}$$

where K is the number of rules, O^i is the class generated by rule i (O^i takes values of 0, 1, ..., $L-1$ where L is the number of classes) and D_p^i was defined in Eq. (6.22). The values of O_p are integers in the range [0, $L-1$].

The following formula is used to determine the class (C_p):

$$C_p = (\text{int})(O_p + 0.5) \tag{6.28}$$

where "int" means taking the nearest smaller integer value.

The centroid defuzzification is commonly used in fuzzy logic control applications and function approximations. It also works well with a two-category classification problem [24]. However, for a classification problem with more than two categories, a large number of fuzzy rules are needed to achieve a satisfactory performance by using this defuzzification scheme [26].

6.6.4 Two-layer Perceptron Approach

Defuzzification parameters can be optimized by training a feedforward neural network [27], [78]. In this scheme, we consider each rule making a contribution to each output class to a certain degree. We define

$$P_{jp} = \sum_{i=1}^{K} w_{ji} D_p^i \tag{6.29}$$

where w_{ji} a weight which indicates the contribution of rule i to class j and P_{jp} represents a degree that a pattern belongs to class j. The winning class is j_* which maximizes P_{jp}. Note that D_p^i is defined in Eq. (6.22).

We can use a two-layer perceptron to determine w_{ji}'s (see Fig. 6.11). The outputs of the network are P_{jp}'s and the number of output nodes is equal to the number of the classes (L). In the network version, we have

$$P_{jp} = f(\text{net}_{jp}) = f(\sum_{i=1}^{K} w_{ji} D_p^i). \tag{6.30}$$

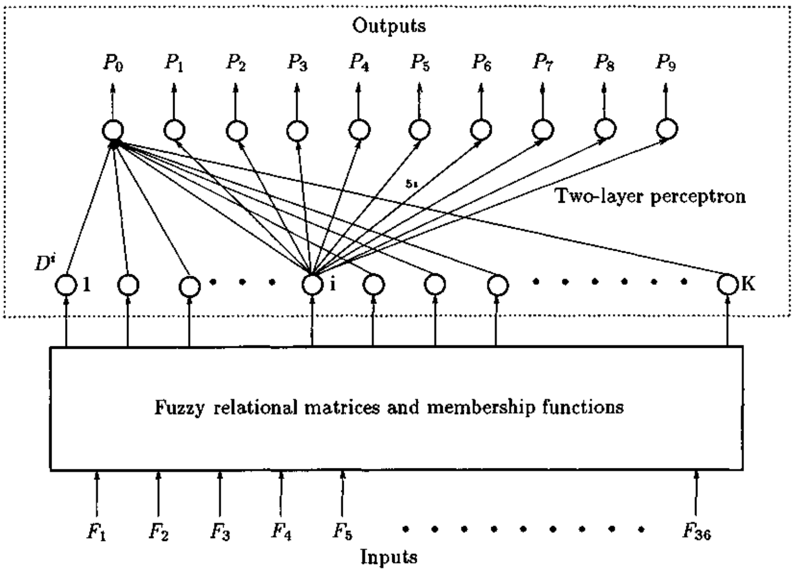

Outputs
P_0 P_1 P_2 P_3 P_4 P_5 P_6 P_7 P_8 P_9
Two-layer perceptron
D^i
1
i
K
Fuzzy relational matrices and membership functions
F_1 F_2 F_3 F_4 F_5 F_{36}
Inputs

Figure 6.11 The two-layer perceptron for optimizing defuzzification parameters of a handwritten digit recognizer (only part of connections are drawn for clarity).

The inputs are D_p^i ($i = 1, 2, ..., K$) and the number of input nodes is equal to the number of fuzzy rules.

A two-layer perceptron is trained using all training patterns to optimize the connection weights by minimizing the error function:

$$E = \sum_{p=1}^{N} E_p = \sum_{p=1}^{N} \frac{1}{2} \sum_{j=0}^{L} (P_{jp} - T_{jp})^2 = \frac{1}{2} \sum_{p=1}^{N} \sum_{j=0}^{L} (P_{jp} - T_{jp})^2 \tag{6.31}$$

where N is the number of training patterns. T_{jp} (0 or 1) and P_{jp} (0.0 to 1.0) are the desired output and the actual output for pattern p, respectively.

Based on the gradient descent algorithm under the batch mode, w_{ji} is modified by

$$\Delta w_{ji} = -\eta \sum_{p=1}^{N} \frac{\partial E_p}{\partial w_{ji}} \tag{6.32}$$

where η is the learning rate and $(\frac{\partial E_p}{\partial w_{ji}})$ is given by

$$\frac{\partial E_p}{\partial w_{ji}} = f'(\text{net}_{jp})(P_{jp} - T_{jp}) D_p^i. \tag{6.33}$$

Substituting Eq. (6.33) into Eq. (6.32), we have

$$\Delta w_{ji} = \eta \sum_{p=1}^{N} f'(\text{net}_{jp})(T_{jp} - P_{jp})D_p^i. \tag{6.34}$$

If the sigmoidal transfer function is used in the output layer, then we have

$$f'(\text{net}_{jp}) = f(\text{net}_{jp})(1 - f(\text{net}_{jp})) = P_{jp}(1 - P_{jp}), \tag{6.35}$$

and

$$w_{ji}(t+1) = w_{ji}(t) + \Delta w_{ji}(t) = w_{ji}(t) + \eta \sum_{p=1}^{N} P_{jp}(1 - P_{jp})(T_{jp} - P_{jp})D_p^i. \tag{6.36}$$

The initial weights w_{ji}'s are set to

$$w_{ji} = \begin{cases} 1 & \text{if rule } i \text{ produces class } j, \\ 0 & \text{otherwise.} \end{cases} \tag{6.37}$$

6.7 Applications

Three applications of fuzzy rules are discussed in the following sections. In Section 6.7.1, fuzzy rules learned from training patterns are used to segment geographic map images using the centroid defuzzification scheme [24]. In Section 6.7.2, we discuss printed upper-case English letter recognition using fuzzy rules and the accumulated matching defuzzification. In Section 6.7.3, handwritten numeral recognition using fuzzy rules produced from a decision tree based technique and the optimized defuzzification based on a two-layer perceptron will be discussed [78].

6.7.1 Segmentation of Map Images

The task is to segment grey scale geographic map images into foreground (characters, roads, streets, boundaries, etc) and background parts. Segmentation was done pixel by pixel using a fuzzy rule-based classification technique.

Features

The map images were scanned into the computer and digitized in R, G and B format with 256 intensity levels for each component. A 256-level grey scale image was then obtained by the following simple transformation:

$$I(i,j) = \frac{R(i,j) + G(i,j) + B(i,j)}{3}. \tag{6.38}$$

As a result, the darkest pixel has grey level 0 and the brightest pixel has grey level 255.

Three features were extracted for map segmentation. The first feature is termed *difference intensity* (DI), which is the difference between a pixel intensity and its local *average intensity* (AI) as measured over a 7×7 region. This feature is defined as

$$DI(i,j) = I(i,j) - AI(i,j) \tag{6.39}$$

where $AI(i,j)$ is defined as

$$AI(i,j) = \frac{\sum_{p=i-3}^{i+3} \sum_{q=j-3}^{j+3} I(p,q)}{49} \tag{6.40}$$

This feature is a measure of relative brightness of a pixel to its neighboring pixels. A foreground pixel is usually darker than the neighbors, so it has a negative value.

The second feature is the local *standard deviation* (SD) as measured over a 7×7 region. This feature is defined as

$$SD(i,j) = \sqrt{\frac{\sum_{p=i-3}^{i+3} \sum_{q=j-3}^{j+3} [I(p,q) - LA(p,q)]^2}{49}}. \tag{6.41}$$

This feature is a measure of homogeneity of a region of which the pixel is in the center. A background pixel usually has a smaller SD value than a foreground pixel but in an area where the map color is changed from one to another, both a background pixel and a foreground pixel have the similar SD values.

The third feature (J) is a measure of the local contrast of a darker pixel against its background. This feature is defined as

$$J(i,j) = \frac{\max[0, B(i,j) - I(i,j)]\text{sgn}[C(i,j)]}{LA(i,j)} \tag{6.42}$$

where sgn[] is the sign operator which was defined in Eq. 2.35. $C(i,j)$ measures the difference of a pixel intensity and the average intensity of eight neighboring pixels shown in Fig. 6.12 and is defined as

$$\begin{aligned} C(i,j) \;=\; & \frac{1}{8}[I(i-3,j) + I(i-2,j) + I(i+2,j) + I(i+3,j) + \\ & I(i,j-3) + I(i,j-2) + I(i,j+2) + I(i,j+3)] - \\ & I(i,j). \end{aligned} \tag{6.43}$$

Note that $C(i,j)$ is different from $DI(i,j)$ defined in Eq. (6.40) in that $C(i,j)$ uses only 8 out of 49 pixels in the 7×7 region to calculate the average intensity and that $I(i,j)$ is the second operand in the subtraction.

$B(i,j)$ is a measure of the average intensity of relative brighter pixels ($C(p,q) \leq 0$) in the 9×9 region and is computed by

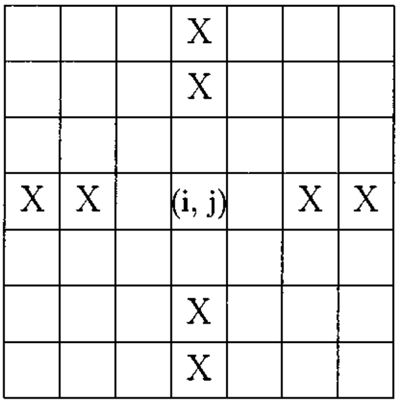


Figure 6.12 Eight neighboring pixels for calculating $C(i,j)$.

$$B(i,j) = \frac{1}{N_b} \sum_{\substack{i-4 \le p \le i+4 \\ j-4 \le q \le j+4 \\ C(p,q) \le 0}} I(p,q) \tag{6.44}$$

where N_b is the number of relative brighter pixels.

A foreground pixel usually has a positive value of J and most background pixels have negative, zero and small positive values. A window having larger size yields a better detection of thicker line patterns while it makes the system difficult to separate very close lines (often resulting in blurred characters). We used a 9×9 region to compute feature J (computing $B(i,j)$) and a 7×7 region to compute LA and SD. Two different window sizes were used here to reduce the line width problem as discussed above.

Membership Functions

Membership functions were derived from 40 c-means clusters with an extension factor $a = 3.0$ and a merging threshold $l_T = 0.08$ (see Fig. 6.13).

Fuzzy Rules

Fuzzy Rules were produced using the technique of learning from examples. We assigned each c-means cluster a label which the majority of feature vectors in that

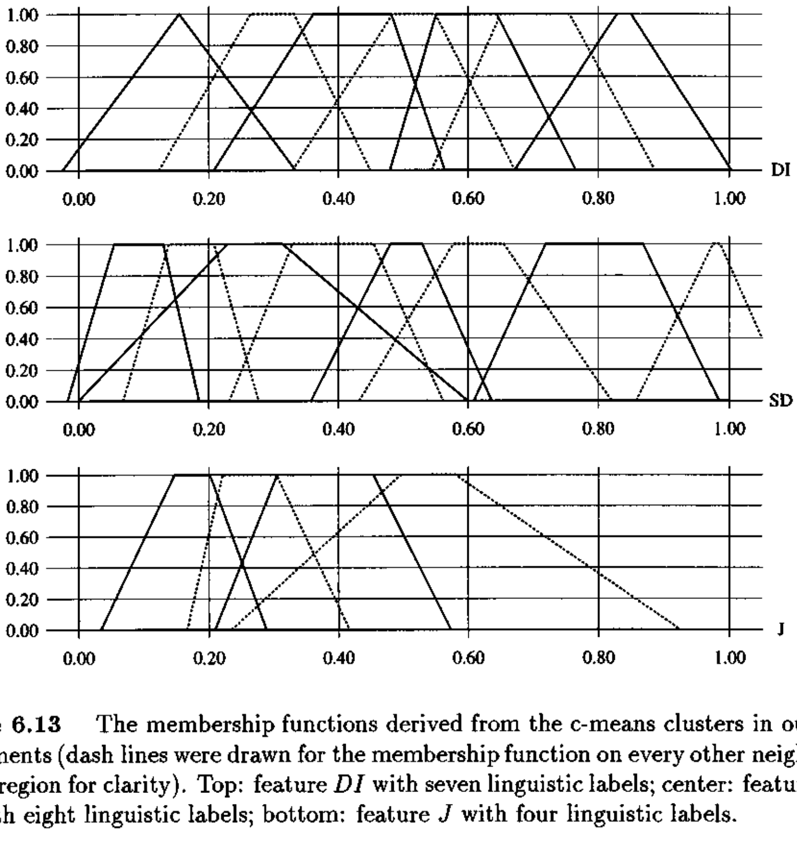


Figure 6.13 The membership functions derived from the c-means clusters in our experiments (dash lines were drawn for the membership function on every other neighboring region for clarity). Top: feature DI with seven linguistic labels; center: feature SD with eight linguistic labels; bottom: feature J with four linguistic labels.

J=R0	R0	R1	R2	R3	R4	R5	R6	R7
R0								
R1								
R2								
R3								
R4	B	B				B	B	
R5			B	B	B	B	B	B
R6						B	B	B

DI

SD

J=R1	R0	R1	R2	R3	R4	R5	R6	R7
R0								
R1								
R2				F				
R3			F	F	F	F	F	
R4		B	F					
R5								
R6								

DI

SD

J=R2	R0	R1	R2	R3	R4	R5	R6	R7
R0								F
R1						F	F	F
R2				F		F	F	
R3					F			F
R4								
R5								
R6								

DI

SD

J=R3	R0	R1	R2	R3	R4	R5	R6	R7
R0								
R1								
R2							F	F
R3						F		
R4			F					
R5								
R6								

DI

SD

Figure 6.14 The fuzzy rule bank for the map image segmentation.

cluster belong to. Forty labeled c-means cluster centers were then used to produce 34 fuzzy rules (see Fig. 6.14). The number of fuzzy rules produced was smaller than the number of clusters due to the fuzzy subsets merging as discussed in Section 2.3.4. Examples of these fuzzy rules are:

Rule 1: IF DI is R_1 AND SD is R_6 AND J is R_2, THEN it is a *foreground* pixel.
Rule 2: IF DI is R_5 AND SD is R_4 AND J is R_0, THEN it is a *background* pixel.

Defuzzification

The centroid defuzzification formula as defined in Section 6.6.3 was used to determine the output for each input pattern (pixel). The output O_p is within [0, 1]. If $O_p \leq 0.5$, the image pixel is classified as a background pixel, otherwise as a foreground pixel.

Table 6.2 A summary of the experiments on fuzzy rule-based map image segmentation (UDTF: the uniformly distributed triangular functions; and KMCV: the membership functions derived from the c-means centers and variances).

Exp.	Membership functions	# Linguistic labels	# Training examples	# Fuzzy rules
FR1	UDTF	4,5,4	80,989 training vectors	38
FR2	UDTF	5,6,5	80,989 training vectors	57
FR3	UDTF	7,8,4	80,989 training vectors	82
FR4	KMCV	7,8,4	40 c-means centers	34

Experiments

The system was tested on an image database of 22 grey scale geographic maps. A computer program has been developed for selecting the training pixels by drawing short lines using the computer mouse on a map image displayed on the computer screen. For each map image, we picked up several thousand background and foreground pixels as training examples (about 2% of total pixels for a typical 450×450 image). Altogether we had 80,989 pixels (42049 background pixels and 38,940 character or line pixels) from these images as training examples. Three features DI, SD and J were then extracted for each of these pixels.

Experimental results on the database of 22 map images showed that the technique achieved good and reliable results. Compared with fuzzy rules based on the uniformly distributed triangular membership functions (see Fig. 2.4) with fuzzy rules learned from all the training examples, our technique has the following two advantages: (1) better performance obtained using a similar number of rules; and (2) fewer rules required to achieve similar performance. Figures 6.15 shows an original map image with a size of 562×447, the segmented image using an adaptive thresholding method, three segmented images using fuzzy rules based on uniformly distributed triangular membership functions (Experiments FR1, FR2 and FR3 summarized in Table 6.2 with 38, 57 and 82 rules, respectively), and the segmented image using the technique based on fuzzy rules derived from the c-means clusters (Experiment FR4 with 34 rules summarized in Table 6.2).

In adaptive thresholding, the image was divided into small blocks with a size of 28×28. A threshold was determined from the histogram of each block. The threshold values were then interpolated to classify each pixel in the image. We can see that the segmentation quality from a fuzzy rule-based technique is better than that obtained by using the adaptive thresholding method. In the segmented image using the adaptive thresholding method, a large number of characters were blurred and boundaries between regions were not correctly detected (resulting in many darker areas). The segmentation quality from fuzzy rules based on the c-means clusters is also similar to that by using more fuzzy rules (34 rules compared with 82) based on

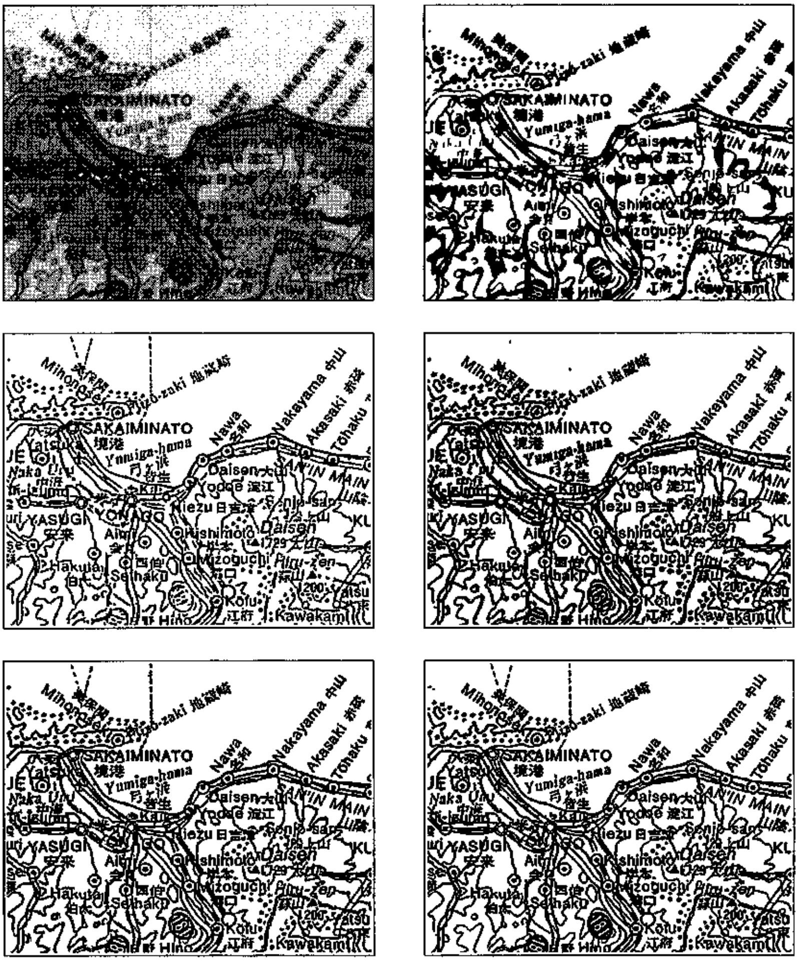


Figure 6.15 Top-left: a grey scale map image; top-right: the segmented map image using the adaptive thresholding method; center-left: the segmented map image using 38 fuzzy rules (Exp. FR1); center-right: the segmented map image using 57 fuzzy rules (Exp. FR2); bottom-left: the segmented map image using 82 fuzzy rules (Exp. FR3); bottom-right: the segmented map image using 34 fuzzy rules derived from the c-means clusters (Exp. FR4).

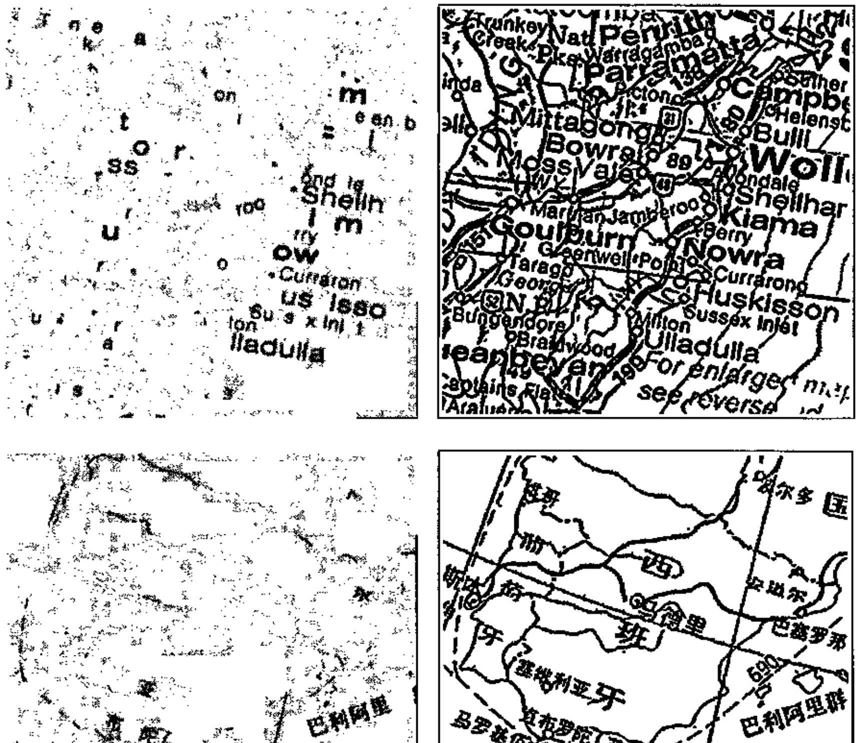


Figure 6.16 Example 1: two original grey scale map images and their corresponding segmented images by using 34 fuzzy rules derived from the c-means clusters (Exp. FR4).

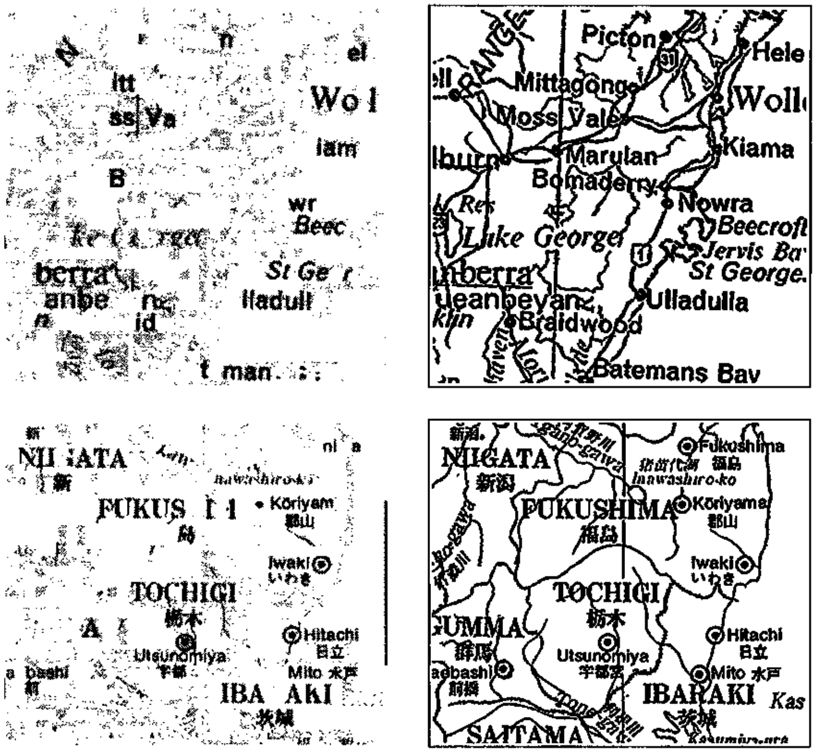


Figure 6.17 Example 2: two original grey scale map images and their corresponding segmented images by using 34 fuzzy rules derived from the c-means clusters (Exp. FR4).

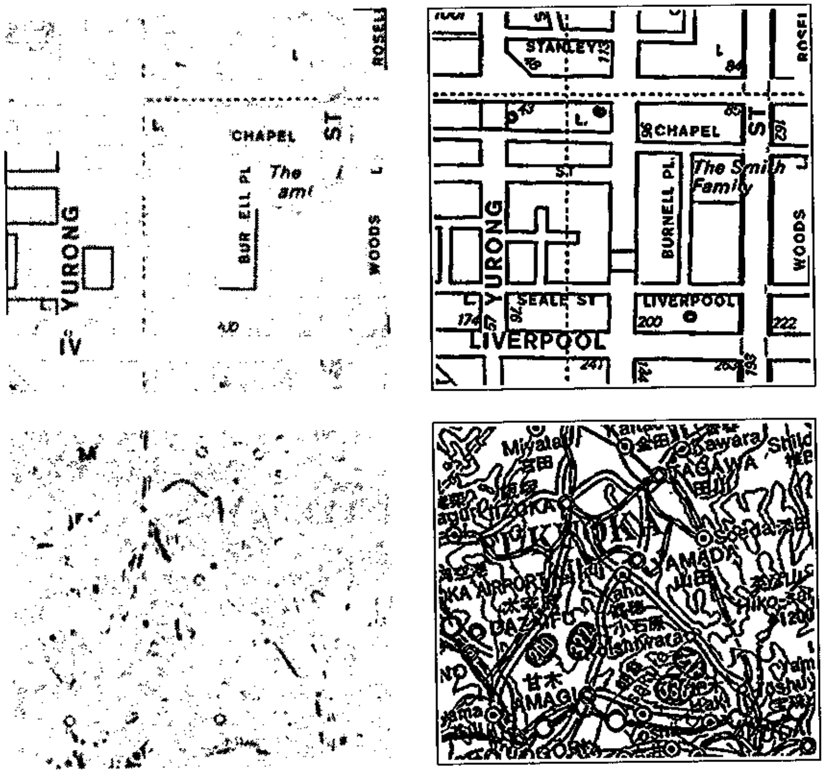


Figure 6.18 Example 3: two original grey scale map images and their corresponding segmented images by using 34 fuzzy rules derived from the c-means clusters (Exp. FR4).

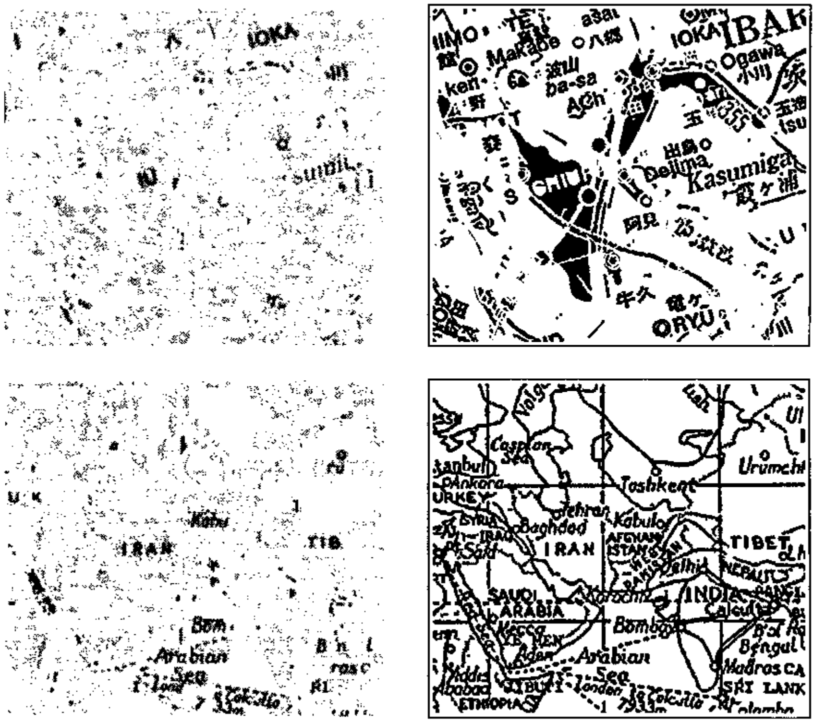


Figure 6.19 Example 4: two original grey scale map images and their corresponding segmented images by using 34 fuzzy rules derived from the c-means clusters (Exp. FR4).

the uniformly distributed triangular membership functions with fuzzy rules learned from all 80,989 training examples. Figure 6.15 shows that the segmented image using 38 fuzzy rules with standard triangular membership functions is too bright and that some parts of characters are missed. The segmented image using 57 fuzzy rules with standard triangular membership functions is too dark, and again some parts of characters are missed. Figures 6.16 to 6.19 show eight map images and their corresponding segmented images using the fuzzy rules derived from the c-means clusters. We can see that the technique achieved a satisfactory performance on these map images.

6.7.2 Printed Upper-case English Letter Recognition

The task is to recognize 26 printed letters of different fonts with or without noise.

Database

We have 512 clean samples for each of the 26 letters and altogether there are 13,312 samples in the data set. We printed the letters with 64 fonts of size point 9 and scanned each character 8 times. Two versions of noisy data sets, test set 2 (Test 2) and test set 3 (Test 3) were then obtained by rotation and adding noise. Figures 6.20 and 6.21 show some of the letters in the clean data set and the noisy letters in test set 3, respectively.

Feature Extraction and Selection

We start with using 113 features as reported in [80]. They include 36 features from four of the 3×3 scaled images obtained from line convolutions between the positive part of the external contour of the image and each of four orientations (horizontal, vertical, and two diagonal directions), and 36 features from four of the 3×3 scaled images obtained from line convolutions between the negative part of the external contour and each of the four orientations. Based on the orientation in which the line convolution is performed, an external contour is divided into the positive and negative parts as shown in Fig. 6.22. The feature set also includes 32 features representing convex and concave curvatures in the 4×4 image. The last nine features represent the number of holes in each of the small blocks in the 3×3 image.

Using the feature selection technique based on a feature entropy measurement with one-dimensional clustering [81], 30 features were selected for the experiment.

Membership Functions

Membership functions were produced from the centers and variances of 26 c-means clusters. We set $a = 4.0$ and $l_T = 0.1$ in the experiment. The numbers of linguistic labels for 30 features are listed in Table 6.3.

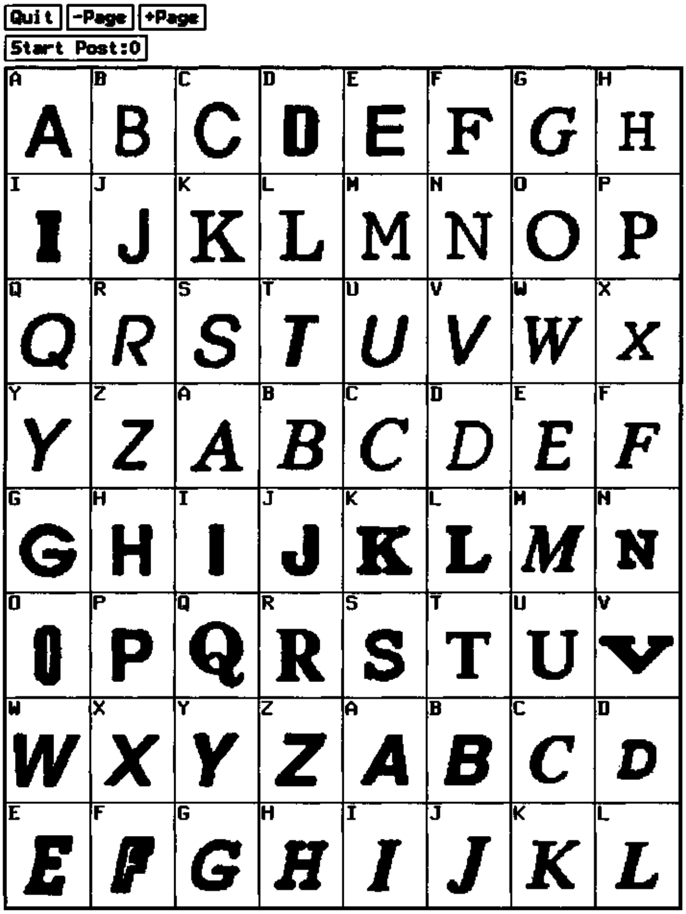


Figure 6.20 Examples of printed upper-case English letters from the clean data set.

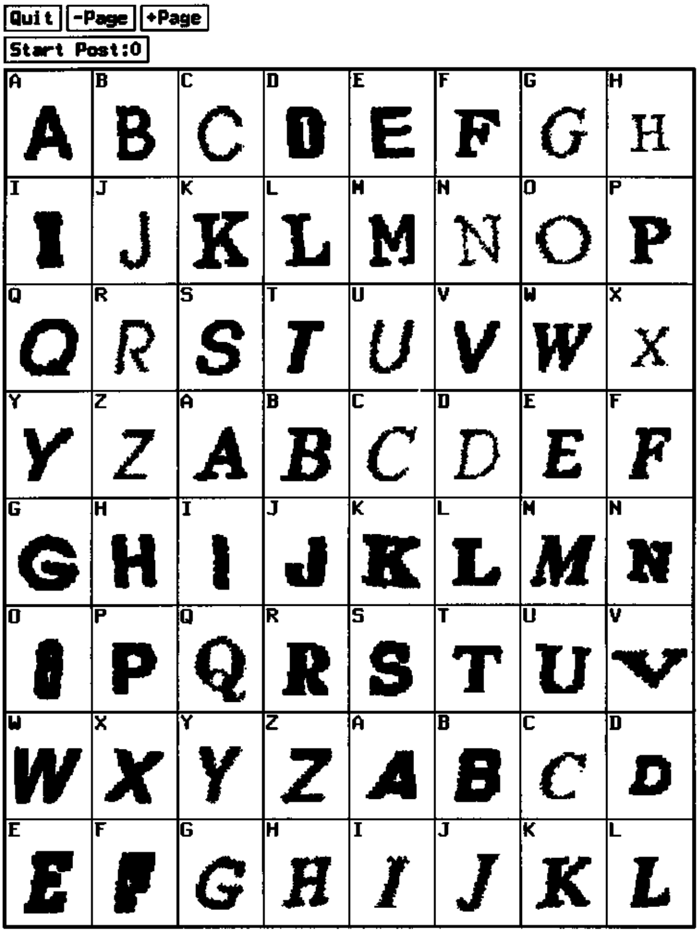


Figure 6.21 Examples of printed upper-case English letters from the noisy data set.

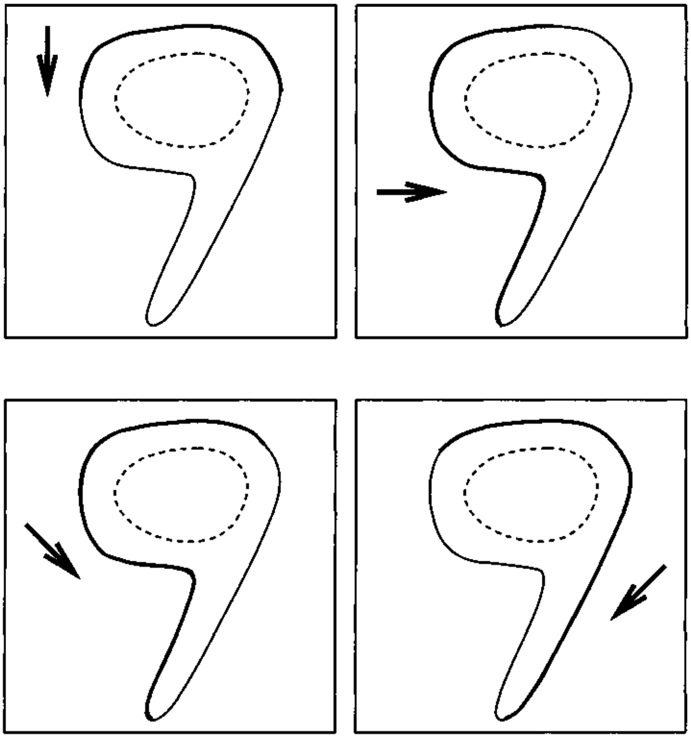


Figure 6.22 The positive (thicker lines) and negative (thinner lines) parts of the external contour of a character image when the line convolution with each of four orientations (horizontal, vertical, and two diagonal directions) is performed.

Table 6.3 The numbers of linguistic labels for 30 features used for printed uppercase English letter recognition.

X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
7	5	7	6	5	6	7	6	6	6
X_{11}	X_{12}	X_{13}	X_{14}	X_{15}	X_{16}	X_{17}	X_{18}	X_{19}	X_{20}
7	7	6	7	5	6	8	8	5	7
X_{21}	X_{22}	X_{23}	X_{24}	X_{25}	X_{26}	X_{27}	X_{28}	X_{29}	X_{30}
6	7	7	6	5	6	7	6	7	7

Fuzzy Rules

Using the supervised c-means clustering algorithm (class by class), we obtained 520 clusters (20 clusters for each letter) for 6,656 clean samples. These clusters were used as examples to produce 520 fuzzy rules by learning from examples.

Defuzzification

The maximum accumulated matching defuzzification was used. For each unlabeled letter, we accumulated the matching degrees of rules with the same class. The final decision was to assign the pattern to the class which had the maximum accumulated matching degree.

Experiments

We randomly split 13,312 clean letter patterns into two data sets with 6,656 patterns in each set. One set was used as training set and the other as test set 1 (Test 1). A system of 520 fuzzy rules achieved 98.8% correct classification on the training set, and 98.5%, 93.0% and 90.8% on Test 1, Test 2 and Test 3, respectively. Note that only clean letter patterns were used for learning the fuzzy rules here. The performance can be improved by using either more features or more training samples.

6.7.3 Handwritten Numeral Recognition

The task is to recognize handwritten digits 0 to 9 using fuzzy rules.

Database

Handwritten numerals we used for experiments were extracted from a database produced by the US National Institute of Standards and Technology (NIST Special Database 3 of handwritten segmented characters). We used 10,426 numerals from a set of images (from f0000 to f0499) for training and another 10,426 numerals from a different set of images (from f0500 to f0999) for testing. The training and test characters were written by different people.

Features

A set of 36 structural features from the skeleton images were used.

(a) *Preprocessing:*

A grey scale character image was first binarized and then thinned using the thinning algorithm as described in [33] (see Fig. 6.23). A set of junction points and tips (end points) were found and segments between these points were traced. Junction points within four pixels are merged. On curved (but non-circular) segments, corner points were identified from the rate of the direction change.

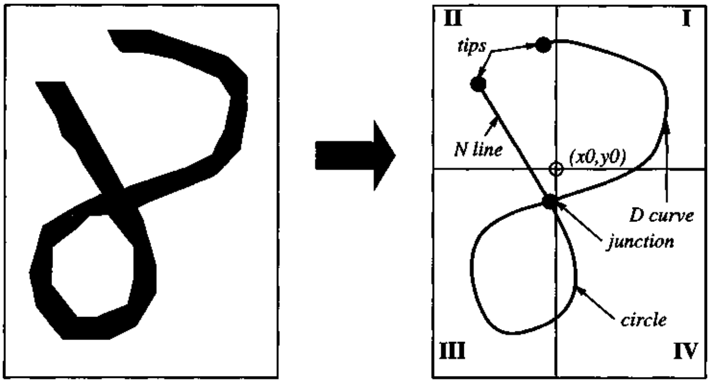


Figure 6.23 An example of thinning and segment representation.

(b) *Labeling:*

We used the fuzzy labeling method proposed by Siy and Chen [82] for our skeleton based handwritten numeral recognition.

A *node set* in the skeleton image is defined as a collection of the *tips* (points that have one neighbor), *corners* (points that have two neighbors and where an abrupt change of line direction occurs), and *junctions* (points that have more than two neighbors).

A *branch* is a segment connecting a pair of adjacent nodes. A *circle* is a special branch, connected to a single node. We categorized each segment as one of 12 types, *H line, V line, P line, N line, C curve, D curve, A curve, V curve, S curve, Z curve, curve* and *circle* (see Fig. 6.24). Type *curve* was reserved for a curve segment which could not be assigned to any of the other six curve types.

The measure of straightness of a branch is determined by fitting it to a straight line using the least squares error method. The straightness of a non-circular branch is defined as

$$f_{SL} = \begin{cases} 1 - S/S_T & \text{if } S < S_T, \\ 0 & \text{if } S \geq S_T \end{cases} \tag{6.45}$$

where S_T is a threshold for the fitting error. A branch is classified as a curve, if $0 \leq f_{SL} < 0.5$, or a straight line, if $0.5 \leq f_{SL} \leq 1$.

According to its angle with the horizontal direction, a straight line segment is classified as *H line, V line, P line* or *N line.* A curved segment is classified as one of the six curve types based on its shape information. If a curved segment cannot be assigned to any of these types, it is considered to be type *curve.*

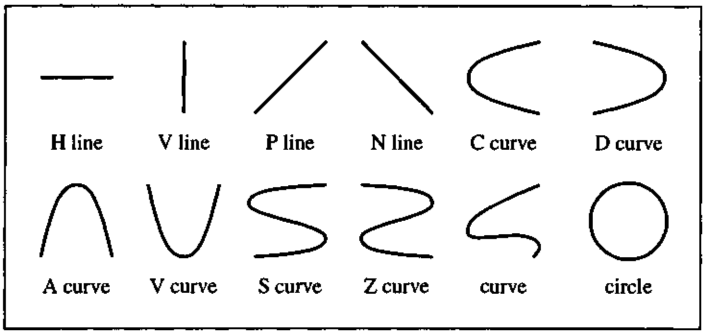


Figure 6.24 Twelve types of line segments.

(c) *Feature Extraction:*

Preliminary examination of the NIST Special Database 3 [83] showed that less than 0.5% of numeral characters had more than six segments, so no more than the six longest segments (in the order of decreasing lengths) were considered for each numeral skeleton image. Four attributes (features) were used to describe a segment. They include:

1. The *type* of a segment;

2. The normalized length of the i-th segment, l_i^n. Suppose that the width and the height of the image (in pixels) are w and h, respectively, the number of segments is N_l (could be greater than six), and the number of pixels in the i-th segment is N_{pi}. We have

$$l_i = \sum_{p=1}^{N_{pi}-1} \text{step}(p, p+1) \tag{6.46}$$

where step$(p, p+1)$ is defined as

$$\text{step}(p,q) = \begin{cases} 1 & \text{if } q \in N_4(p) \\ \sqrt{2} & \text{if } q \in N_8(p) \wedge \neg(q \in N_4(p)) \end{cases} \tag{6.47}$$

where p is a character pixel on the skeleton, and $N_4(p)$ and $N_8(p)$ are the 4-neighbors and the 8-neighbors of p, respectively.

The normalized l_i, denoted by l_i^n, is defined as

$$l_i^n = \frac{l_i}{2(w+h)}. \tag{6.48}$$

3. x_{ci}^n and y_{ci}^n are the normalized horizontal and vertical coordinates, respectively, of the relative center of the i-th segment to the center of the skeleton image, (x_g, y_g). Firstly, the center of the i-th segment, (x_{ci}, y_{ci}), is calculated by

$$x_{ci} = \frac{\sum_{p=1}^{N_{p_i}-1} \mathrm{step}(p, p+1)\frac{x_p + x_{p+1}}{2}}{l_i}, \tag{6.49}$$

$$y_{ci} = \frac{\sum_{p=1}^{N_{p_i}-1} \mathrm{step}(p, p+1)\frac{y_p + y_{p+1}}{2}}{l_i}. \tag{6.50}$$

Secondly, the center of the skeleton image, (x_g, y_g), is given by

$$x_g = \frac{\sum_{i=1}^{N_l} l_i x_{ci}}{\sum_i l_i}, \tag{6.51}$$

$$y_g = \frac{\sum_{i=1}^{N_l} l_i y_{ci}}{\sum_i l_i}. \tag{6.52}$$

Finally, x_{ci}^n and y_{ci}^n are defined as

$$x_{ci}^n = \frac{x_{ci} - x_g}{w}, \tag{6.53}$$

$$y_{ci}^n = \frac{y_{ci} - y_g}{h}. \tag{6.54}$$

The most important feature for numeral recognition using the syntactic approach is the *type*. Length is also useful in discriminating between numerals which have one or more segments of similar type, such as digits "2" and "3" (both have *D curves*). Center information was used to approximate the position of a segment.

Besides the four features of all six segments, which make 24 features, the number of segments which a digit has (set the maximum value to six) is used as a feature because some digits tend to have more segments than the others. Also used are features including the numbers of end points in each of the four domains I, II, III, and IV, the normalized total length, the center coordinates of the skeleton digit image, the numbers of straight lines, curves and circles, and the aspect ratio of the image. All together 36 features are used in this study. For a non-existing segment j ($j > N_l$ and $N_l < 6$), we set the *type* to "none", $l_i^n = 0$, $x_{ci}^n = -x_g/w$, and $y_{ci}^n = -y_g/h$.

Table 6.4 Membership grades for the number of segments.

number	1	2	3	4	5	6
1	1.0	0.25	0.1	0.0	0.0	0.0
2	0.25	1.0	0.25	0.1	0.0	0.0
3	0.1	0.25	1.0	0.25	0.1	0.0
4	0.0	0.1	0.25	1.0	0.25	0.1
5	0.0	0.0	0.1	0.25	1.0	0.25
6	0.0	0.0	0.0	0.1	0.25	1.0

Membership Functions

Fuzzy grade matrices were constructed for symbolic or discrete-valued features. They can be determined heuristically. In our experiments, we constructed the grade matrices heuristically for the features "number" and the features "type".

Table 6.4 shows the fuzzy grade matrix used for the number of segments. Note that we only made use of the six longest segments so that the values of the feature are in the range from 1 to 6. The columns represent the actual number of the segments in a skeletal numeral image, and the rows indicate the number of segments required by the antecedent conditions. The fuzziness in the number of the segments was introduced, because: (1) we deleted segments shorter than 10 pixels although they may be useful for classification to a certain degree; and (2) there was a threshold used in finding the corners of a long curved segment and as a result, a false corner could be introduced or a genuine corner could be missed out. As can be seen from Table 6.4, the values of 0.25 was assigned to the nearest neighbor(s) and 0.1 to the second nearest neighbor(s) of an element. Similarly, fuzzy grade matrices were generated for the other discrete-valued features.

Table 6.5 shows a fuzzy grade matrix for the type of a segment. There are 12 types used for our skeleton based handwritten numeral recognition. The column represents the actual type of a segment of a numeral (note that some fuzzy measure was already introduced in preprocessing) and the row is the type which the antecedent conditions require. The fuzziness for the segment types is introduced, because: (1) fuzziness was used in assigning a type to a segment; (2) there are similarities between some types; and (3) human recognition is very robust in dealing with segment types.

For features with continuous values, the membership functions as defined in Eqs. (6.13) and (6.14) were used, in which a was set to 0.5.

Fuzzy Rules

Fuzzy rules were produced using the technique based on a decision tree learning algorithm. A set of 151 fuzzy rules were produced from 10,426 training samples.

Table 6.5 Membership grades for segment types: HL (H line), VL (V line), PL (P line), NL (N line), CC (C curve), DC (D curve), AC (A curve), VC (V curve), SC (S curve), ZC (Z curve), CU (curve), and CI (circle).

type	HL	VL	PL	NL	CU	CC	DC	AC	VC	SC	ZC	CI
HL	1.0	0.0	0.25	0.25	0.25	0.1	0.1	0.25	0.25	0.1	0.1	0.0
VL	0.0	1.0	0.25	0.25	0.25	0.25	0.25	0.1	0.1	0.1	0.1	0.0
PL	0.25	0.25	1.0	0.0	0.25	0.25	0.25	0.25	0.25	0.1	0.1	0.0
NL	0.25	0.25	0.0	1.0	0.25	0.25	0.25	0.25	0.25	0.1	0.1	0.0
CU	0.1	0.1	0.1	0.1	1.0	0.5	0.5	0.5	0.5	0.5	0.5	0.1
CC	0.0	0.25	0.25	0.25	0.5	1.0	0.0	0.25	0.25	0.25	0.25	0.25
DC	0.0	0.25	0.25	0.25	0.5	0.0	1.0	0.25	0.25	0.25	0.25	0.25
AC	0.25	0.0	0.25	0.25	0.5	0.25	0.25	1.0	0.0	0.25	0.25	0.25
VC	0.25	0.0	0.25	0.25	0.5	0.25	0.25	0.0	1.0	0.25	0.25	0.25
SC	0.1	0.1	0.1	0.1	0.25	0.25	0.25	0.25	0.25	1.0	0.5	0.1
ZC	0.1	0.1	0.1	0.1	0.25	0.25	0.25	0.25	0.25	0.5	1.0	0.1
CI	0.0	0.0	0.0	0.0	0.1	0.25	0.25	0.25	0.25	0.1	0.1	1.0

Defuzzification

A two-layer perceptron was trained to perform the optimized defuzzification. The perceptron has 152 input nodes, each for a rule plus a bias node, and 10 output nodes.

Experiments

The NIST Special Database 3 was used for our experiments. We extracted 10,426 samples for training and another 10,426 samples from different forms for testing.

Table 6.6 lists the correct classification rates obtained from the decision trees and rules, and fuzzified decision rules. A classifier using a decision tree with 2,163 rules achieved 98.8% correct classification on the training set and 91.4% on the test set. After pruning to 828 rules, the classifier had a slightly higher rate on the test set (91.9%) with a cost of training accuracy down from 98.8% to 97.1%. Using the simplified decision rules (151 rules), we achieved 94.6% and 90.7% for the correct classification rates on the training set and the test set, respectively. It is seen that the performance degrades due to the simplification in extracting the rules from the decision tree. However, after the membership grades were introduced into the rules and the optimized defuzzification applied, the fuzzified decision rules achieved 97.7% and 95.0% for the correct classification rates on the training set and the test set, respectively. This is a significant improvement over the decision trees and the simplified decision rules. Figure 6.25 shows the examples of handwritten digits which are correctly classified by the fuzzified decision rules.

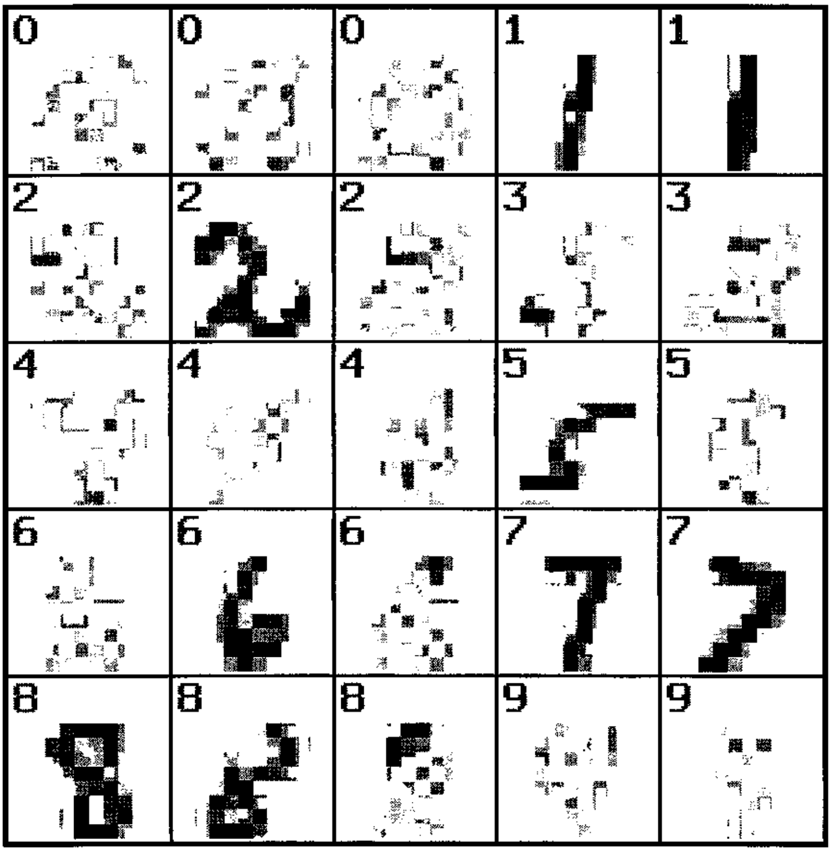
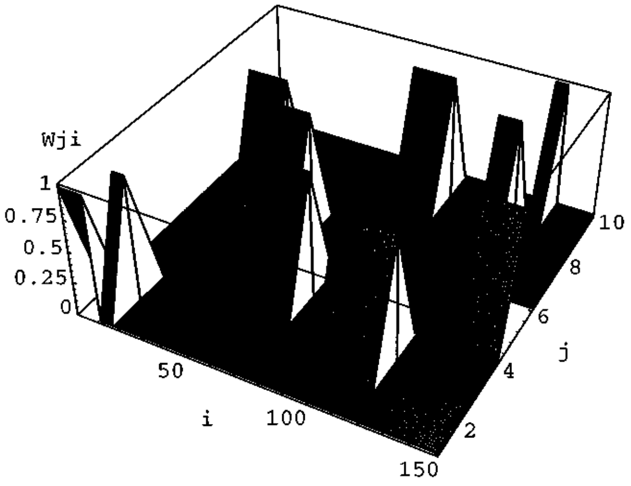


Figure 6.25 Examples of handwritten digits which are correctly classified by fuzzified decision rules.



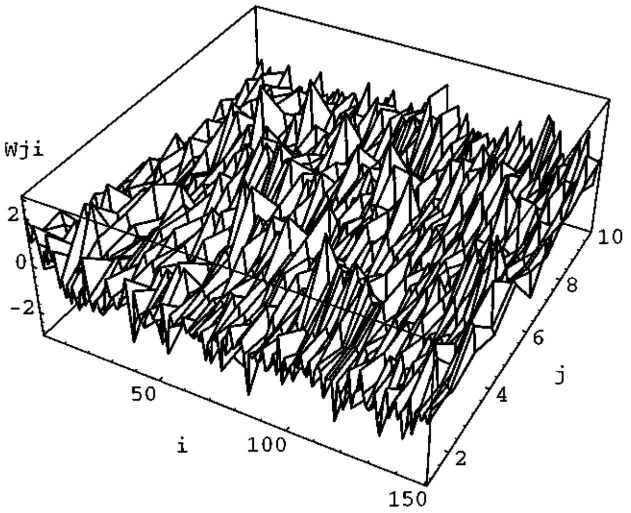


Figure 6.26 Rule weights ($w_{ji}, 1 \leq i \leq 151$ for 151 fuzzy rules, $1 \leq j \leq 10$ for ten classes) before (top) and after optimization (bottom).

Table 6.6 Classification performance for different techniques used.

	Performance (%)	
Techniques	On training set	On test set
Decision tree (before pruning, size of the tree: 2,163)	98.8	91.4
Decision tree (after pruning, size of the tree: 828)	97.1	91.9
Simplified decision rules (151 rules)	94.6	90.7
Fuzzified decision rules (151 rules)	97.7	95.0

Figure 6.26 shows the weight patterns before and after optimized defuzzification. We can view a positive weight as an excitatory signal and a negative weight as an inhibitory signal. Now a rule does not response to only one class since it has positive weights to some classes and negative weights to other classes. Also a class receives contributions from all rules which can be either excitative or inhibitive.

6.8 Concluding Remarks

We discussed in this chapter three techniques for producing fuzzy rules from numerical data, Wang and Mendel's learning from examples, the decision tree based approach, and the neural network approach proposed by Krishnapuram *et al.*

Generating fuzzy rules using the approach of learning from examples is the most straightforward. It does not require intensive computation because it only performs a simple one-pass operation on the training data. However, the system performance is very much dependent of the partition of the input and output domain intervals and the chosen membership functions. Therefore, it is strongly suggested that one of clustering techniques as discussed in Section 2.3 is used to generate a set of membership functions which actually reflects the real data distribution. This is particularly important for pattern recognition and image processing where an open loop system is usually adopted. Another problem with this approach is that we tend to have a large number of fuzzy rules when a great number of training samples are available. The problem can be partially solved by either using a clustering procedure to obtain the centers of clusters which are then used to produce a smaller set of fuzzy rules (see Section 6.5.1) or utilizing a fuzzy rule minimization technique as one discussed in Section 6.5.2 to reduce the number of produced fuzzy rules.

Fuzzy rules derived from decision trees are flexible in the sense that different rules tends to use different features and have different lengths. Therefore, these rules are more similar to those extracted from expert knowledge. In this approach, selection of membership functions is simple and a compact set of fuzzy rules can be easily obtained by using a well-defined procedure for simplifying the decision rules. Another advantage of the approach is that the ID3 learning algorithm can deal with mixed types of inputs, symbolic, discrete-valued, and continuous-valued, which is important for high level image processing and pattern recognition. Compared with Wang and Mendel's approach of learning from examples, the approach based on decision trees takes longer learning time.

Neural networks have found many applications in various optimization problems. A neural network approach seems to be a natural way to produce an optimized set of fuzzy rules. However, the progress in this direction has been less promising so far. Fuzzy aggregation networks which can be used to produce fuzzy rules has the difficulty in the learning convergence because of the exponent parameters (p and $1/p$) used in each node which are very sensitive to small perturbations. Therefore, only a small network can be trained successfully. As a result, only a very small number of rules can be produced, which is not suitable for a practical application.

In this chapter, we have also discussed a few defuzzification schemes for fuzzy rule-based pattern recognition, including maximum matching, maximum accumulated matching, centroid defuzzification, and the two-layer perceptron approach. Among them, the two-layer perceptron is the most promising approach because it incorporates the parameter optimization into the defuzzification process.

To design a fuzzy rule-based recognition system, the following steps have to be taken:

1. A training set has to be constructed.

2. A suitable set of membership functions have to be chosen or to be generated from the training data.

3. A learning technique has to be used to produce a set of fuzzy rules from the training data or the cluster centers obtained from a clustering algorithm.

4. A defuzzification scheme has to be adopted to perform the classification and recognition.

Unfortunately, all of the aforementioned tasks are still more or less problem dependent. Furthermore, the classifier performance strongly depends on the classification power of the extracted features. Three applications described in Section 6.7 provide us a general guide for designing a fuzzy rule-based recognition system.

Chapter 7

Combined Classifiers

7.1 Introduction

By using different feature sets and different classification techniques, classification systems have different performance. Most classifiers have particular strengths and weaknesses in that they can reliably distinguish between some patterns, but may confuse others. There are several ways to reduce this confusion by using extra features, tuning classifier parameters, and combining complementary classifiers.

Suppose that there are K individual classifiers $(f_k, k = 1, 2, ..., K)$. For an unlabeled input pattern $\mathbf{x}$ each f_k will assign $\mathbf{x}$ a label $\omega_k \in \Lambda \cup \{\phi\}$ where $\Lambda = \{1, 2, ..., L\}$ (L is the number of classes) representing a pattern class and ϕ denoting the rejection. No matter what classifier structure it has and what theory and methodology it is based on, f_k is regarded as a function which receives an input $\mathbf{x}$ then outputs a class label ω_k and probably the degrees to which the pattern belong to different classes. A combined classifier is to make use of all the results from K classifiers, $f_k(\mathbf{x})$, to make a final classification. Therefore, building a combined classifier is equivalent to build a function $F(\mathbf{x}) = \omega$, $\omega \in \Lambda \cup \{\phi\}$, based on $f_k(\mathbf{x})$, $k = 1, 2, ..., K$, such that a better classification performance would be obtained.

Xu *et al* [84] distinguished three classifier types according to the amount of information returned. Class 1 classifiers return only the selected class or rejection. Class 2 classifiers return a ranked list of classes, and Class 3 classifiers return a measurement associated with each class. Combination techniques including voting [85], [86], maximizing posterior probabilities [84], and Dempster-Shafer evidence theory [87], [88] have been proposed to combine Class 1 classifiers. Borda counts and logistic regression have been used to combine Class 2 classifiers. Results from both Class 2 and 3 classifiers can be combined using rule-based, weighted-sum, associative switch [89], trained perceptron, and fuzzy integral techniques [90], [91], [92]. A hierarchical decision making based on various recognition experiences can always be used to combine different classifiers [26], [93]. In this chapter, we will discuss several techniques which combine complementary classifiers to improve the classification performance. An application of these combination classifiers on handwritten numeral character recognition will be reported following the general discussion on the combination techniques.

In the discussion of the following sections, we will use the correct classification rate (r_c), rejection rate (r_r), and substitution rate (r_s) to measure the classification performance of a classifier. Assume that n_r patterns are rejected from a total of n patterns to be classified and among those classified patterns, n_c patterns are correctly classified. We can define r_r, r_c, and r_s as

$$\begin{aligned} r_r &= \frac{n_r}{n}, && (7.1)\\ r_c &= \frac{n_c}{n - n_r}, && (7.2)\\ r_s &= \frac{n - n_r - n_c}{n - n_r} && (7.3)\\ &= 1 - r_c. && (7.4) \end{aligned}$$

To get a better idea about the classification and misclassification of each individual class, the confusion matrix is often adopted. A generalized confusion matrix, P, is defined as

$$P = (n_{ij}) = \begin{bmatrix} n_{11} & n_{12} & \dots & n_{1L} & n_{1(L+1)} \\ n_{21} & n_{22} & \dots & n_{2L} & n_{2(L+1)} \\ \vdots & \vdots & & \vdots & \vdots \\ n_{L1} & n_{L2} & \dots & n_{LL} & n_{L(L+1)} \end{bmatrix} \qquad (7.5)$$

where L is the number of classes, n_{ij} $(i, j = 1, 2, ..., L)$ denotes the number of patterns from class i being classified to digit j, and $n_{i(L+1)}$ $(i = 1, 2, ..., L)$ is the number of patterns from class i being rejected. Obviously, we have

$$\begin{aligned} n_r &= \sum_{i=1}^{L} n_{i(L+1)}, && (7.6)\\ n_c &= \sum_{i=1}^{L} n_{ii}, && (7.7)\\ n &= \sum_{i=1}^{L} \sum_{j=1}^{L+1} n_{ij}. && (7.8) \end{aligned}$$

7.2 Voting Schemes

There are various voting schemes which can pick up a label ω from the labels produced by K classifiers $\{\omega_1, \omega_2, ..., \omega_K\}$.

Unanimous Voting

Accept a classification only if all classifiers produce the same label (not rejection).

$$\omega = \begin{cases} \omega_1 & \text{if } \omega_1 = \omega_2 = \ldots = \omega_K \neq \phi, \\ \phi & \text{otherwise.} \end{cases} \tag{7.9}$$

This combination scheme usually produces a very low substitution rate but with the cost of a high rejection rate.

Majority Voting

If more classifiers label a sample to one class than to any others, then the sample is assigned to that class. Otherwise, it is rejected. Let Ω_c be a subset consisting of all $\omega_k \neq \phi$.

$$\omega = \begin{cases} \omega_k & \text{if } \omega_k \in \Omega_c \text{ and } \forall \omega_{k_1} \neq \omega_k \text{ such that } r(\omega_k) > r(\omega_{k_1}), \\ \phi & \text{otherwise} \end{cases} \tag{7.10}$$

where $r(\omega_k)$ denotes the number of the classifiers which take the value ω_k in Ω_c. Note that the scheme defined in Eq. (7.10) is not the only one for majority voting. There are many variations depending on the problem, the theory and methodology used in each classifier, and the number of classifiers.

Compared with the unanimous voting scheme, this combination method produces a rather low rejection rate but a higher substitution rate.

In general, both voting methods cannot perform a classification without rejection even each individual classifier can do it. Both voting methods can be adopted to combine all types of classifiers.

7.3 Maximum Posteriori Probability

Assume that n_{ij}^k denotes the number of class i patterns which are classified to class j by classifier k ($k = 1, 2, ..., K$). Let $p_k(i|j)$ denote the probability that a pattern, which classifier k assigns to class j, actually belongs to class i. We have

$$p_k(i|j) = \frac{n_{ij}^k}{\sum_{l=1}^{L} n_{lj}^k}. \tag{7.11}$$

We define the overall probability that a sample belongs to class i ($i = 1, 2, ..., L$) as

$$p(i) = \sum_{k=1}^{K} p_k(i|i_k) \tag{7.12}$$

where i_k is the label which classifier k outputs. The output of the combined classifier is i_* which maximizes $p(i)$, that is,

$$p(i_*) = \max_{i=1}^{L} p(i). \tag{7.13}$$

If individual classifiers can perform the classification without rejection, this combination method will guarantee to perform a classification without rejection. The method can be used to combine all types of classifiers.

7.4 Multilayer Perceptron Approach

A Class 3 classifier can provide not only the classification labels but also the measures of likelihood. Usually, the winner-take-all strategy is used for a Class 3 classifier to perform the classification. Let o_i^k be the ith output of classifier f_k. If $o_j^k \geq o_l^k$ for $l = 1, 2, ..., L$ and $l \neq j$, where L is the number of classes, then the output of classifier f_k is ω_j.

A multilayer neural network can be used to combine several Class 3 classifiers to improve the classification performance, in particular, to resolve the conflict among the classifiers (see Fig. 7.1). The number of inputs to the network will be $K \times L$ (K is the number of classifiers) and the number of the outputs will be L. The training patterns are (**y**, **o**), where $\mathbf{o} = [o_1^d, o_2^d, ..., o_L^d]$ are the desired outputs and $\mathbf{y} = [o_1^1, o_2^1, ..., o_i^k, ..., o_L^K]^T$, where o_i^k is the ith output of classifier k ($i = 1, 2, ..., L$, and $k = 1, 2, ..., K$). If the pattern is from class j, then the desired output is

$$o_l^d = \begin{cases} 1 & \text{if } l = j, \\ 0 & \text{otherwise.} \end{cases} \tag{7.14}$$

The multilayer neural network is trained by using a backpropagation algorithm with the mean square error function defined as

$$E_p = \frac{1}{2} \sum_{j=1}^{L} \left[o_j^d(\mathbf{y}_p) - o_j(\mathbf{y}_p) \right]^2 \tag{7.15}$$

where $o_j(\mathbf{y}_p)$ is the actual output for node j of the output layer. We can consider the training of the combination network as a second-order learning following the first-order learning of each individual classifier. The combination classifier can usually achieve a better classification performance. By setting different thresholds in the test mode, we can obtain various correct classification rates over different rejection rates. For an application where the misclassification is vital like a medical system, we can set a threshold in such a way that 100% correct classification can be achieved with the cost of a quite high rejection rate.

7.5 Fuzzy Measures and Fuzzy Integrals

Randomness and fuzziness are two different types of uncertainties. The theory of probability deals with the first type and the theory of fuzzy sets deals with the

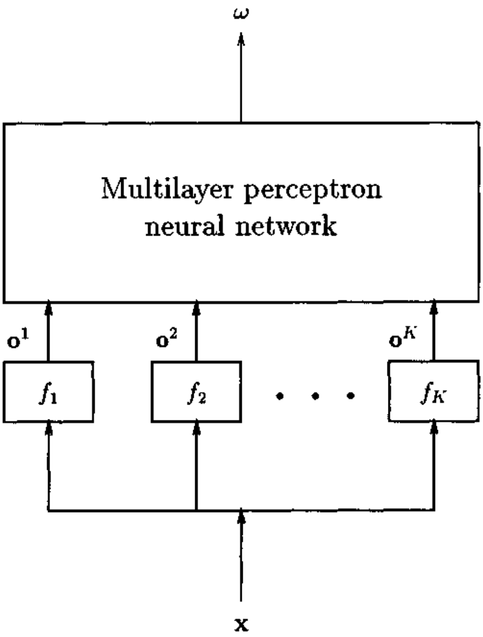


Figure 7.1 A combination classifier using a multilayer perceptron.

second. Stemming from the concept of fuzzy-set theory [1], fuzzy measure and fuzzy integral were proposed by Sugeno [20]. In the same way that probability measure is used as a scale for randomness, fuzzy measure is used as a scale to express the grade of fuzziness. Based on the properties of fuzzy measure, fuzzy integral is an aggregation operator on multi-attribute fuzzy information. Fuzzy integral can be viewed as a fuzzy expectation when compared with statistical expectation, and as a nonlinear integral in comparison with Lebesque integral. The properties of fuzzy measure and the operations on Sugeno and Choquet fuzzy integrals shall be briefly described below.

7.5.1 Fuzzy Measures

Let X be a non-empty set, and $\mathcal{B}$ be a Borel field of X. A set function g: $\Omega \rightarrow$ [0,1] defined on $\mathcal{B}$ is a fuzzy measure if it satisfies the following three axioms:

1. Boundary conditions: $g(\emptyset) = 0$, $g(X) = 1$.
2. Monotonicity: $g(A) \leq g(B)$ if $A \subset B$, and $A, B \in \mathcal{B}$.
3. Continuity: $\lim g(A_i) = g(\lim A_i)$ if $A_i \in \mathcal{B}$ and A_i is monotone increasing.

A g_λ-fuzzy measure is also proposed by Sugeno [20] which satisfies another condition known as the λ-rule ($\lambda > -1$):

$$g(A \cup B) = g(A) + g(B) + \lambda\, g(A)g(B) \tag{7.16}$$

where $A, B \subset X$, and $A \cap B = \emptyset$.

Let $X = \{x_1, x_2, ..., x_n\}$ be a finite set, and $g_i = g(\{x_i\})$, $i = 1, ..., n$, be the values over the set of singletons of X (g_i is now called a fuzzy density function of the g_λ measures), then $g_\lambda(A)$ can be expressed by [92]

$$\begin{aligned} g_\lambda(A) &= \sum_{i=1}^{n} g_i + \lambda \sum_{i_1=1}^{n-1} \sum_{i_2=i_1+1}^{n} g_{i_1} g_{i_2} + \ldots + \lambda^{n-1} g_1 g_2 \ldots g_n \\ &= \frac{1}{\lambda}\left[\prod_{x_i \in A}(1+\lambda g_i) - 1\right], \ \lambda \neq 0 \end{aligned} \tag{7.17}$$

Recall that $g(X) = 1$ and if the fuzzy densities g_i are known, then the g_λ-measures can be identified by solving the following equation for λ:

$$\lambda + 1 = \prod_{i=1}^{n}(1 + \lambda g_i) \tag{7.18}$$

7.5.2 Fuzzy Integrals

Let $(X, \mathcal{B}, g)$ be a fuzzy measure space and $f : X \to [0,1]$ be a measurable function. We also define $X = \{x_1, x_2, ..., x_n\}$; and $0 \leq f(x_1) \leq f(x_2) ... \leq f(x_n) \leq 1$, (if not, the elements of X are rearranged to hold this relation), and $A_i = \{x_i, x_{i+1}, ..., x_n\}$. Then, the Sugeno integral over $A \subset X$ of the function f with respect to a fuzzy measure g is defined by:

$$\int_A f \circ g = \vee_{i=1}^{n} [f(x_i) \wedge g(A_i)] \tag{7.19}$$

where $\wedge$ denotes "minimum", and $\vee$ stands for "supremum".

The Choquet integral, which is another form of fuzzy integrals, of f with respect to g is defined by:

$$\int_A f \, dg = \sum_{i=1}^{n} [(f(x_i) - f(x_{i-1}))g(A_i)] \tag{7.20}$$

in which $f(x_0) = 0$.

There are a number of interpretations on the meaning of fuzzy integrals. A fuzzy integral can be understood as a fuzzy expectation [20], the maximal grade of agreement between two opposite tendencies [94], or the maximal grade of agreement between the objective evidence and the expectation [92]. In this paper, a fuzzy integral is considered as a maximum degree of belief (for a class or an object) obtained from the fusion of several objective evidences where the importance of multiple attributes are subject to fuzzy measures.

7.5.3 A Fuzzy Integral Model for Classifier Fusion

The algorithm for the fusion of the results from multiple classifiers is described as below.

1. *Forming the confusion matrix*: Let the confusion matrix P be the results of correctly classified and misclassified patterns obtained from the training set (see Section 7.1). It is established for each classifier and expressed in the form:

$$P^k = (n_{ij}^k) \tag{7.21}$$

   where k is one of the K-classifiers ($k = 1, 2, ..., K$), and P^k is an $L \times L$ matrix (L is the number of classes). Note that we only examine the non-rejection classifier here. For $i = j$, n_{ij}^k indicates the number of patterns from class i being correctly classified by the k-classifier; whereas $i \neq j$, n_{ij}^k indicates the number of patterns from class i being misclassified as class j.

2. *Computing the initial fuzzy densities*: The initial fuzzy density used here is in fact calculated as a discrete probability, and interpreted as the degree in which

a classifier identifies a certain class correctly. This initial fuzzy density can be defined as

$$g_i^k = \frac{n_{ii}^k}{\sum_{j=1}^{L} n_{ij}^k} \tag{7.22}$$

in which $0 < g_i^k < 1$ is the fuzzy density of the class i with respect to the k-classifier.

3. *Computing the correction factors*: It is always observed that each classifier is more robust than the others in classifying the patterns from some classes, but also more error-prone in the classification of the patterns from other classes than the other classifiers. Therefore, it is necessary to take into account the effect of the relation between the frequencies of correct and incorrect classifications. This can be done by introducing two sets of correction factors which then be used to adjust the initial fuzzy density as defined by Eq. (7.18). The first correction set can be expressed by the following equation

$$\delta_{ij}^k = \begin{cases} 1 & for \quad i = j \\ \dfrac{n_{ii}^k - n_{ij}^k}{n_{ii}^k} & for \quad i \neq j \end{cases} \tag{7.23}$$

where δ_{ij}^k is the correction factor for the fuzzy density g_i^k, which counts for the patterns from class i being misclassified as class j.

The second correction set is

$$\gamma_{ij}^k = \begin{cases} 1 & for \quad n_{ij}^k \leq n_{ij}^l \\ \dfrac{n_{ij}^l}{n_{ij}^k} & for \quad n_{ij}^k \geq n_{ij}^l \\ 0.0001 & for \quad n_{ij}^k = 0 \end{cases} \tag{7.24}$$

where n_{ij}^l, $l \neq k$, is the number of class i being misclassified as class j by the l-classifier.

4. *Finding classes having highest score from all classifiers*:

$$N = \{(V_i)_{max}^k\} \tag{7.25}$$

where N is the vector of the classes V_i having highest score given by the *k-th* classifier.

5. *Updating the initial fuzzy densities*:

   The initial fuzzy densities are updated using the correction factors in Eqs. (7.23) and (7.25) by the following equation:

$$g_i^{*k} = g_i^k \times (\delta_{ij}^k \times \cdots \times \delta_{ik}^k)^{w_1} \times (\gamma_{ij}^k \times \cdots \times \gamma_{ik}^k)^{w_2}. \tag{7.26}$$

   where g_i^{*k} is the updated fuzzy density, the subscripts j and k are the indices given by $(V_i)_{max}$ as defined in Eq. (7.25) by the other classifiers, w_1 and w_2 are the exponential weights assigned to the first and the second sets of the correction factors as defined by Eqs. (7.23) and (7.24) respectively.

6. *Computing fuzzy measures and fuzzy integrals*: Having determined the fuzzy densities g_i^{*k}, the g_λ- fuzzy measures for each class from different classifiers can be obtained using Eq. (7.16). Then the fuzzy integral can be computed accordingly using Eqs. (7.19) or (7.20) where the function f is the fuzzy membership function mapped from the actual output of the test set. This membership function is an S-function which is defined by Zadeh [1] as

$$S(x) = \begin{cases} 2x^2 & for \quad 0 \le x \le 0.5 \\ 1 - 2(x-1)^2 & for \quad 0.5 \le x \le 1 \end{cases} \tag{7.27}$$

   where $x \in [0, 1]$ is the output from the test set by the classifier.

7. *Decision making of the final class*: The final output class from the combination classifier is made by selecting the class with the highest integrated value.

7.6 Applications

In this application, different methods are used to combine three classifiers using different methodologies and different feature sets for handwritten numeral recognition. The same training and test sets as reported in Section 6.7.3 are used. There are 10,426 digit patterns in the training set and 10,426 digit patterns written by different people in the test set. Classifier 1 is based on the skeleton representation of characters and the ID3-derived fuzzy rules (see Fig. 7.2) which was reported in Section 6.7.3. The classification performance of Classifier 1 on the training and test sets are shown in Tables 7.1 and 7.2, respectively. Note that we adopted a table which is similar to the confusion matrix except that the rejection rates are not shown and instead the column of "rate" is added which shows the correct classification rates for each individual class and the overall performance as well. Different thresholds set in the test mode resulted in the substitution/rejection performance of Classifier 1 which are displayed in Fig. 7.7. The second classifier we used is Yan's optimized nearest neighbor classifier [95] which makes use of image intensities only and will be briefly discussed in Section 7.6.1. Classifier 3 makes use of Markov chains which are obtained

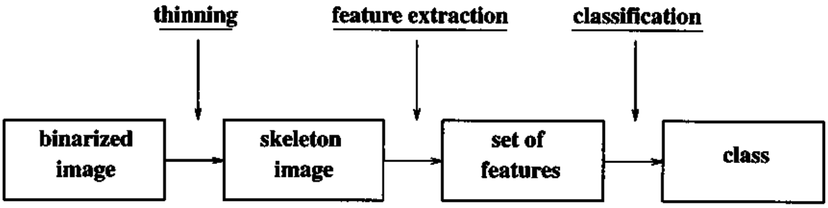


Figure 7.2 Block diagram of the skeleton based character recognition system.

Table 7.1 Classification performance of the ID3-derived fuzzy rule-based classifier on the training set.

digit	0	1	2	3	4	5	6	7	8	9	rate (%)
0	1059	0	1	4	2	1	1	0	13	0	97.96
1	0	1166	0	0	0	1	4	2	0	0	99.40
2	0	1	985	7	9	3	5	9	1	2	96.38
3	1	1	8	1031	2	9	2	9	0	1	96.90
4	1	0	4	2	1000	1	0	2	3	5	98.23
5	1	0	2	8	2	897	1	0	3	3	97.82
6	0	3	3	0	0	4	1010	0	4	1	98.54
7	0	1	5	3	3	2	0	1072	0	1	98.62
8	3	0	3	2	5	3	4	1	990	18	96.21
9	1	3	1	4	11	3	0	5	4	978	96.83
overall											97.72

from contour images of characters. In this technique, a numeral contour is traversed in a well-defined order and the Markov chain is used to perform sequential analysis and to match with the models. The Markov chain-based technique for handwritten character recognition is discussed in Section 7.6.2.

7.6.1 Optimized Nearest Neighbor Classifier

The nearest neighbor rule has been studied in pattern recognition for many years. For a basic nearest neighbor classifier (NNC), all training samples are used as prototypes and an input sample is assigned to the class of the closest prototype. Although the classification rule is conceptually very simple, it has a robust performance. Theoretically, its asymptotic classification error is bounded above by twice the Bayes error [96]. However, the NNC has not been used widely in the past because a large memory storage space and long computing time are needed for its implementation if the number of training samples is large. Many techniques have been developed in the past to

Table 7.2 Classification performance of the ID3-derived fuzzy rule-based classifier on the test set.

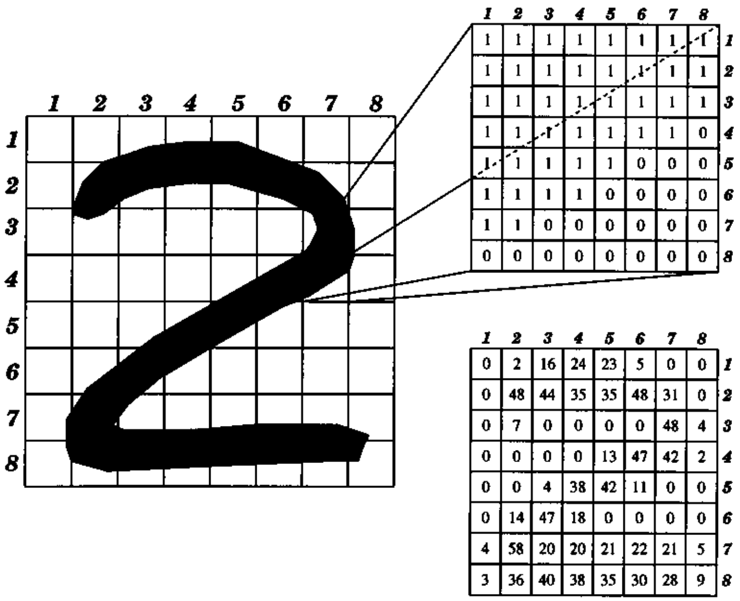
digit	0	1	2	3	4	5	6	7	8	9	rate (%)
0	1041	0	3	3	1	8	11	2	13	2	96.03
1	0	1138	0	0	0	2	5	2	2	0	99.04
2	1	5	945	30	15	13	7	16	3	0	91.30
3	2	3	26	996	5	25	2	7	0	5	93.00
4	0	6	2	2	992	7	0	5	8	12	95.94
5	1	7	6	9	6	857	14	1	5	5	94.07
6	16	12	14	0	2	7	986	1	11	0	93.99
7	0	2	12	8	6	1	0	1046	0	2	97.12
8	8	1	2	4	10	6	6	1	960	22	94.12
9	6	0	1	5	14	9	0	11	12	938	94.18
overall											94.95

overcome the computational complexity problem of the NNC by deleting noisy prototypes or combining similar ones, but as a result of these operations, the recognition performance is usually degraded even for a small reduction of prototypes [96].

Recently Yan has developed a method for building a computationally efficient NNC with high classification power [95], [97]. In this method, a clustering technique combines training samples to produce a small number of prototypes. The prototype feature values are mapped to the weights of a multi-layer perceptron neural network which is then trained using all available training data. The weights of the trained perceptron are mapped back to a set of new and optimized prototypes. After learning, the optimized prototypes have the same classification performance as the multi-layer neural network used for training. The new nearest neighbor classifier can be efficiently implemented since the number of optimized prototypes is small and the classification procedure can be sped up using various ranking strategies [95], [97].

Yan's optimized nearest neighbor classifier makes use of 64 intensities as inputs. Each digit character is first rescaled and centered onto 64×64 grids. The number of character pixels in each 8×8 local window are counted and the 8×8 gray scale images are obtained. The resulting 64 intensities in the image are in the range of [0, 64] and are used as features. Figure 7.3 shows an example of the 64×64 binary image and 8×8 gray scale handwritten numeral image for the digit 2.

Tables 7.3 and 7.4 show the classification performance of Yan's optimized nearest neighbor classifier on the training and test sets, respectively.



	1	2	3	4	5	6	7	8	
	1	1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1	2
	1	1	1	1	1	1	1	1	3
	1	1	1	1	1	1	1	0	4
	1	1	1	1	1	0	0	0	5
	1	1	1	1	0	0	0	0	6
	1	1	0	0	0	0	0	0	7
	0	0	0	0	0	0	0	0	8

1	2	3	4	5	6	7	8	
0	2	16	24	23	5	0	0	1
0	48	44	35	35	48	31	0	2
0	7	0	0	0	0	48	4	3
0	0	0	0	13	47	42	2	4
0	0	4	38	42	11	0	0	5
0	14	47	18	0	0	0	0	6
4	58	20	20	21	22	21	5	7
3	36	40	38	35	30	28	9	8

Figure 7.3 Left: an example of the 64 × 64 binary image of the handwritten numeral 2; top-right: one of zoom-in 8 × 8 blocks with binary codes; bottom-right: the 8 × 8 gray scale values of the handwritten numeral 2 on the left.

Table 7.3 Classification performance of the optimized nearest neighbor classifier on the training set.

digit	0	1	2	3	4	5	6	7	8	9	rate (%)
0	1076	0	0	0	0	2	2	0	1	0	99.54
1	0	1159	5	3	0	0	1	1	3	1	98.81
2	1	1	1001	3	0	1	2	3	10	0	97.95
3	2	0	7	1034	0	8	1	1	6	5	97.18
4	0	0	1	0	1000	0	1	2	1	13	98.23
5	1	0	3	12	1	897	0	0	3	0	97.82
6	6	1	0	0	2	2	1011	0	2	1	98.63
7	0	3	0	0	4	0	0	1068	2	10	98.25
8	2	4	1	4	3	5	2	0	1006	2	97.76
9	0	1	0	0	3	0	0	3	1	1002	99.21
overall											98.35

Table 7.4 Classification performance of the optimized nearest neighbor classifier on the test set.

digit	0	1	2	3	4	5	6	7	8	9	rate (%)
0	1076	0	1	1	0	4	1	0	1	0	99.26
1	0	1147	0	0	0	0	0	0	2	0	99.83
2	5	0	1021	2	0	1	2	2	2	0	98.65
3	0	0	6	1035	0	14	0	1	14	1	96.64
4	0	0	0	0	1023	1	3	0	1	6	98.94
5	1	0	0	8	1	886	5	0	10	0	97.26
6	2	1	0	0	0	2	1042	1	1	0	99.33
7	0	0	2	0	6	0	0	1056	1	12	98.05
8	8	8	1	21	3	15	2	2	956	4	93.73
9	3	0	3	8	10	4	0	8	9	951	95.48
overall											97.77

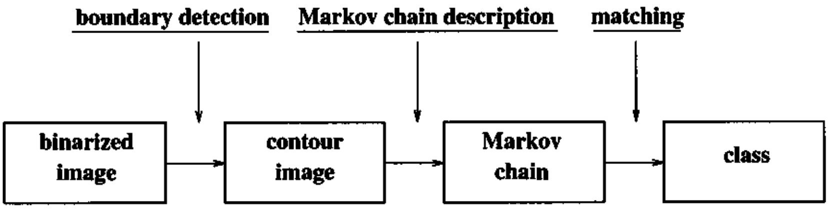


Figure 7.4 Block digram of the contour based character recognition system.

7.6.2 Markov Chain-based Classifier

Mathematical Description

Figure 7.4 shows the block diagram of the contour based character recognition system. The Markov chain digit recognition system is based on a discrete time, discrete state Markov system. At time t, the system occupies state S^t, one of n_s possible states. In the designed Markov system, state transition probabilities depend only on the current state, i.e.:

$$P(S^t) = P(S^t|S^{t-1}). \tag{7.28}$$

At each time step, another discrete quantity is measured, which is called the "observable" O^t. The probability distribution for the n_o observable values depends on the current state of the system:

$$P(O^t) = P(O^t|S^t). \qquad (7.29)$$

Thus a sample image may be described by a Markov chain, which is an ordered list of observations

$$C = [S^0, O^0], \ldots, [S^t, O^t], [S^{t+1}, O^{t+1}], \ldots, [S^N, O^N]. \qquad (7.30)$$

Each of the ten digits is described by a Markov model λ, with n_s states and n_o observable feature values. Each model is defined by three probability distributions:

1. the probability that the system will initially be in state i,

$$\pi_i = P(S^0 = i), \qquad 1 < i < n_s \qquad (7.31)$$

2. the probability that, given a system in state i, the observed value will be k,

$$b_{i,k} = P(O^t = k|S^t = i), \qquad 1 < i < n_s, \quad 1 < k < n_o \qquad (7.32)$$

3. the probability that a system in state i will be in state j at the next observation,

$$a_{i,j} = P(S^t = j|S^{t-1} = i), \qquad 1 < i,j < n_s \qquad (7.33)$$

Given a particular Markov chain, the probability that it was produced by the system with Markov model λ is

$$P(C|\lambda) = \pi_{S^0} b_{S^0,O^0} [\prod_{t=1}^{N} a_{S^{t-1},S^t} b_{S^t,O^t}]. \qquad (7.34)$$

The recognition strategy is that when a test digit is presented to the system, its Markov chain is derived, and the probability that the chain was produced by each of the ten models calculated. The sample is assigned to the class of the model with the highest probability value.

Feature Selection and Chain Generation

This section describes the definition of states and observables used in the system, and generation of the Markov chain of a sample image.

Assuming the character image is a square of side 64 pixels, it is split into 16 regions regarded as separate states. Each state is a square of side 16 pixels, located as shown in Fig. 7.5 on the left. The direction from one contour point to the next is treated as the observable, and assigned a discrete value according to the sectors in Fig. 7.5 on the right.

The Markov chain for an image, which is a sequence of state and observable values, is generated by the following steps:

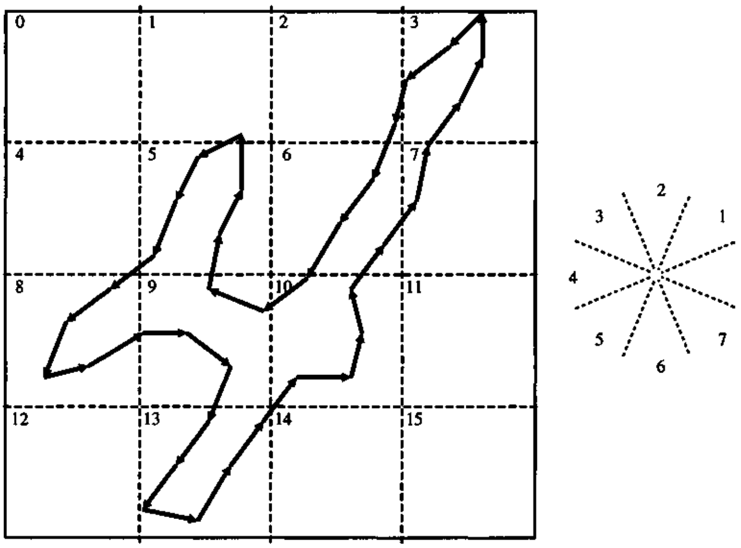


Figure 7.5 A sample digit, showing the external contour traced by arrows, and the 16 sections (states) into which the image was separated. At the right is a compass flower, showing the values assigned to the arrow directions (observable values).

1. Scale the image to fit onto a 64 by 64 region without distortion.
2. Trace the contour of the figure by traversing it in a counter-clockwise manner, starting at the lower left corner.
3. Generate the contour chain by selecting every fifth contour pixel.
4. Generate the state sequence from the locations of each contour chain point.
5. Generate the observable sequence from the direction from one contour chain point to the next.

Model Parameters

Ten Markov models, one for each digit, were generated by deriving probability distributions π, $b_{i,k}$ and $a_{i,j}$ from the same training sample of 10,426 images as those used for the fuzzy ID3 classification. For each training sample, the Markov chain was generated, and the incidence of each combination of observable value and state transition recorded. These incidences were totalized over all samples in each class, to produce probability distributions for the classes.

The ten model system ignores some potential problems with the Markov chain method, including digits which may be written in two completely different ways (for example "4"), digits which are not bound by a single external contour, and the substantial variation occurring if a break occurs in the written stroke (as in the "8" of Fig. 6.23).

Classification of a test sample involves assigning it to the class of the model which is most likely to generate the Markov chain of the sample. If two models produce very similar values of probability, the classification may be regarded as uncertain, so the ratio R of the highest probability to the next highest probability could be used to reject such samples. If the ratio is below some threshold, the sample would be rejected.

We used ten models, one for each digit, created by determining the probability distributions for initial state, state transition and observable value from the 10,426 image training set. The classifier correctly recognized 94.8 % of training set images (see Table 7.5), and 94.2 % of the test set (see Table 7.6) A series of threshold tests resulted in the substitution/rejection performance displayed in Fig. 7.7; rejection occurred if the ratio between the top two probabilities was less than a threshold.

These results indicate that the Markov chain method is not suitable for independent use, but it may be combined with other classifiers for improved performance.

7.6.3 Combined Classifiers

Using different feature sets such as intensities and those based on skeleton and contour images, and different classification methodology such as fuzzy rules, optimized

Table 7.5 Classification performance of the Markov chain-based classifier on the training set.

digit	0	1	2	3	4	5	6	7	8	9	rate (%)
0	1047	5	2	1	1	0	8	3	14	0	96.85
1	2	1150	1	0	2	0	9	2	5	2	98.04
2	3	3	954	18	10	5	1	17	10	1	93.35
3	5	0	20	1008	0	11	0	9	6	5	94.74
4	0	3	4	0	978	1	7	0	12	13	96.07
5	1	0	4	6	0	881	3	7	7	8	96.07
6	6	10	1	0	5	2	988	2	11	0	96.39
7	4	7	9	2	7	4	0	1018	5	31	93.65
8	39	8	9	3	12	14	11	3	909	21	88.34
9	3	1	4	0	16	6	0	10	18	952	94.26
overall											94.81

Table 7.6 Classification performance of the Markov chain-based classifier on the test set.

digit	0	1	2	3	4	5	6	7	8	9	rate (%)
0	1042	8	4	4	0	5	5	1	15	0	96.13
1	1	1133	0	0	1	0	3	1	9	1	98.61
2	3	3	969	17	9	2	3	20	7	2	93.62
3	0	0	16	1014	0	20	2	8	5	6	94.68
4	1	3	4	0	989	4	8	1	12	12	95.65
5	2	1	4	18	0	843	17	1	19	6	92.54
6	4	22	2	1	7	3	998	4	8	0	95.14
7	1	6	19	5	3	1	0	1016	3	23	94.34
8	27	16	11	8	7	12	9	2	913	15	89.51
9	8	0	2	16	23	6	0	16	23	902	90.56
overall											94.18

Table 7.7 Classification performance of the unanimous voting combination classifier on the test set (10.26% of rejection rate).

digit	0	1	2	3	4	5	6	7	8	9	rate (%)
0	1005	0	0	0	0	1	0	0	0	0	99.90
1	0	1123	0	0	0	0	0	0	0	0	100.00
2	0	0	900	0	0	0	0	0	0	0	100.00
3	0	0	0	944	0	0	0	0	0	0	100.00
4	0	0	0	0	951	0	0	0	0	0	100.00
5	0	0	0	1	0	794	1	0	0	0	99.75
6	0	1	0	0	0	0	945	1	0	0	99.79
7	0	0	1	0	2	0	0	997	0	0	99.70
8	0	0	0	1	0	0	0	0	836	0	99.88
9	0	0	0	0	1	0	0	1	0	850	99.77
overall											99.88

prototypes, and Markov chains, the three classifiers aforementioned are complementary to one another to some extent. A better performance would be expected if we combine them using the combination techniques as discussed in this chapter.

Unanimous Voting

If all of the three classifiers produce the same class, we accept the classification. Otherwise, the input pattern is rejected. The combination classifier achieves 99.88% correct classification on the test set (only 13 out of 9,356 patterns were misclassified) after 10.26% of test patterns (1,070 patterns) have been rejected. The resulting confusion matrix is shown in Table 7.7.

Majority Voting

For this three-classifier combination case, the combined classifier perform the classification if two or three classifiers produce the same class. Otherwise, the input pattern is rejected. The combination classifier achieves 98.86% correct classification on the test set after 1.23% of test patterns have been rejected. Shown in Table 7.8 is the confusion matrix obtained by this combination classifier.

Maximum Posteriori Probability

In this application, $p_k(i|j)$ ($k = 1,2,3$ and $i,j = 0,1,...,9$) are obtained from the resulting confusion matrices of individual classifiers on the training set (see Tables 7.1, 7.3, and 7.5). The combination classifier achieves a correct classification rate of 98.38% without rejection on the test set. Table 7.9 shows the confusion matrix of this combination classifier.

Table 7.8 Classification performance of the majority voting combination classifier on the test set (1.23% of rejection rate).

digit	0	1	2	3	4	5	6	7	8	9	rate (%)
0	1072	0	1	0	0	1	0	0	1	0	99.72
1	0	1146	0	0	0	0	0	0	1	0	99.91
2	0	0	1002	3	1	1	0	4	1	0	99.01
3	0	0	6	1039	0	5	0	2	3	1	98.39
4	0	0	0	0	1020	0	2	0	0	3	99.51
5	1	0	0	6	1	888	3	0	3	0	98.45
6	0	4	0	0	0	1	1034	1	1	0	99.33
7	0	0	9	0	3	0	0	1049	1	3	98.50
8	3	2	0	2	2	4	0	1	978	5	98.09
9	3	0	0	6	3	0	0	7	6	953	97.44
overall											98.86

Table 7.9 Classification performance of the maximum posteriori probability combination classifier on the test set.

digit	0	1	2	3	4	5	6	7	8	9	rate (%)
0	1080	0	1	0	0	1	0	0	2	0	99.63
1	0	1148	0	0	0	0	0	0	1	0	99.91
2	2	2	1017	5	1	2	1	4	1	0	98.26
3	0	1	10	1048	0	5	0	2	4	1	97.85
4	0	0	0	0	1024	1	3	1	2	3	99.03
5	1	4	0	6	1	889	7	0	3	0	97.59
6	2	4	0	0	0	1	1038	1	3	0	98.95
7	0	0	9	0	5	0	0	1059	1	3	98.33
8	3	4	0	3	2	4	1	3	994	6	97.45
9	3	0	1	8	5	2	0	7	10	960	96.39
overall											98.38

Table 7.10 Classification performance of the multilayer perceptron combination classifier on the test set.

digit	0	1	2	3	4	5	6	7	8	9	rate (%)
0	1079	0	1	1	0	1	1	0	1	0	99.54
1	0	1147	0	0	0	0	0	0	2	0	99.83
2	2	1	1022	3	0	0	3	1	3	0	98.74
3	0	3	8	1051	0	3	0	0	4	2	98.13
4	0	0	0	0	1019	1	1	4	4	5	98.55
5	1	0	0	6	1	894	2	0	6	1	98.13
6	3	4	1	2	1	2	1035	1	0	0	98.67
7	0	0	5	3	2	0	0	1064	1	2	98.79
8	3	0	1	4	0	2	0	0	1007	3	98.73
9	4	0	0	11	1	1	0	8	9	962	96.59
overall											98.70

Trained Multilayer Perceptron

We use a 31-10-10 perceptron shown in Fig. 7.6 to combine the three classifiers. Thirty inputs are all in the range [0.0, 1.0], which consist of the ten normalized outputs of the defuzzified ID3-derived fuzzy rules, the ten normalized reciprocals of Euclidean distance measures from the optimized nearest neighbor classifier, and the ten normalized logarithms of the outputs of the Markov chain method. The extra input node is used for bias term to the hidden nodes. The perceptron was trained by using the backpropagation algorithm with the same training data we used to produce the fuzzy rules and the optimized defuzzification parameters, to generate the optimized prototypes, and to construct the Markov Chain models. In the training, a tolerance of 0.3 is adopted for the output while in the test, winner-take-all strategy is used to determine the output. The substitution/rejection performance of this combination classifier shown in Fig. 7.7 is obtained by setting different thresholds in the test mode. The combination classifier achieves a correct classification rate of 98.7% without rejection on the test set. The resulting confusion matrix is shown in Table 7.10. As can be seen from the experimental results, the multilayer perceptron combination classifier performs better than each individual classifier.

Fuzzy Integral Approach

The fuzzy-integral fusion classifier is applied to combine the results obtained from three classifiers: the ID3-derived fuzzy rules, the Markov chain, and the optimized nearest neighbor.

After the classifying process, each classifier generates a numerical weight in the

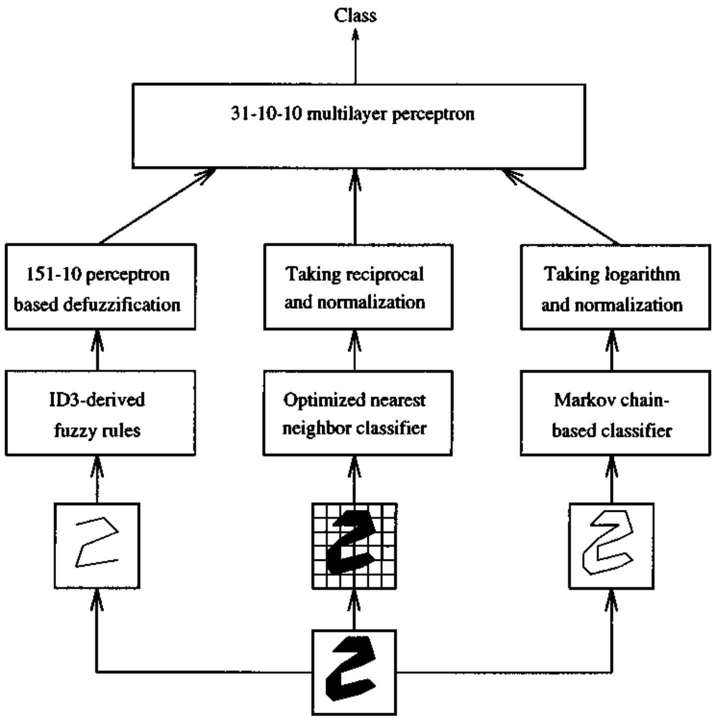

Class
31-10-10 multilayer perceptron
151-10 perceptron based defuzzification
Taking reciprocal and normalization
Taking logarithm and normalization
ID3-derived fuzzy rules
Optimized nearest neighbor classifier
Markov chain-based classifier

Figure 7.6 The multilayer perceptron combination classification system.

Table 7.11 Classification performance of the fuzzy integral combination classifier on the test set.

digit	0	1	2	3	4	5	6	7	8	9	rate (%)
0	1081	0	1	0	0	2	0	0	0	0	99.82
1	0	1149	0	0	0	0	0	0	0	0	100.00
2	6	0	1024	0	0	0	1	2	2	0	99.13
3	0	0	6	1046	0	11	0	2	5	1	97.67
4	0	0	0	0	1025	1	1	0	1	6	99.13
5	0	0	0	4	2	897	5	0	3	0	98.46
6	6	1	0	0	0	2	1039	1	0	0	99.05
7	0	0	2	0	6	0	0	1058	1	10	98.24
8	8	3	0	1	4	12	0	2	986	4	96.67
9	5	0	1	6	6	2	0	5	7	964	96.79
overall											98.49

range of 0 to 1 to each class where 0 indicates an absolute certainty that the sample does not belong to that class, and 1 indicates an absolute certainty that the sample belongs to that class. The local and global performance rates obtained by the three classifiers using the training data set and the test set are as shown in Tables 7.1 to 7.6. The fuzzy-integral fusion classifier as described in Section 7.5 was then applied to combine the three results by making use of these numerical weights and the confusion matrixes. This fuzzy fusion method improves on the results obtained by the three separate classifiers. Table 7.11 shows the combined result on the test set by the fuzzy integral approach. Using $w_1 = 0.9$, and $w_2 = 0.05$ as the exponential weights for the two sets of correction factors, the overall performance rate is increased to 98.76% for the fuzzy-integral combination of the training sets, and to 98.49% for the test set in comparison with 97.77% as the maximum rate obtained by the optimized nearest neighbor classifier.

Performance of Various Combination Methods

In summary, Table 7.12 lists the correct classification rates of individual classifiers and Table 7.13 shows the classification performance of various combination classifiers on the training and test sets. As we can see from the tables, all combination classifiers have better performance than any individual classifier used alone. If a false classification is vital as in medical applications, unanimous voting could be a good choice. Examining into the substitution/rejection relationship for different combination classifiers shown in Fig. 7.7, we found that both the multilayer perceptron and the fuzzy integral model have a better overall performance than each individual classifier. We can also see that the fuzzy integral approach can achieve as good performance as the

Table 7.12 Classification performance of individual classifiers (the numbers in the brackets are the rejection rates).

Techniques	Performance (%)	
	On training set	On test set
ID3-derived fuzzy rules	97.72 (0.0)	94.95 (0.0)
Optimized nearest neighbor classifier	98.35 (0.0)	97.77 (0.0)
Markov chain-based classifier	94.81 (0.0)	94.18 (0.0)

Table 7.13 Classification performance of combined classifiers of using different techniques (the numbers in the brackets are the rejection rates).

Techniques	Performance (%)	
	On training set	On test set
Unanimous voting	99.97 (7.59)	99.88 (10.26)
Majority voting	99.35 (0.67)	98.86 (1.23)
Maximum posteriori probability	99.04 (0.0)	98.38 (0.0)
Multi-layer perceptron	99.83 (0.0)	98.60 (0.0)
Fuzzy integral	98.76 (0.0)	98.49 (0.0)

multilayer perceptron method.

7.7 Concluding Remarks

In this chapter, various techniques including unanimous and majority voting, maximizing posterior probability, trained multilayer perceptron, and fuzzy integral have been proposed for combining two or more independent classifiers in order to achieve a better classification performance. Experimental results on handwritten digit recognition using the combination of the three classifiers, ID3-derived fuzzy rules-based classifier, Markov chain-based classifier, and Yan's optimized nearest neighbor classifier, show that all of the proposed combination techniques can improve the classification performance to some extent. Among them, the multilayer perceptron combination classifier and the fuzzy integral combination classifier achieved the best results.

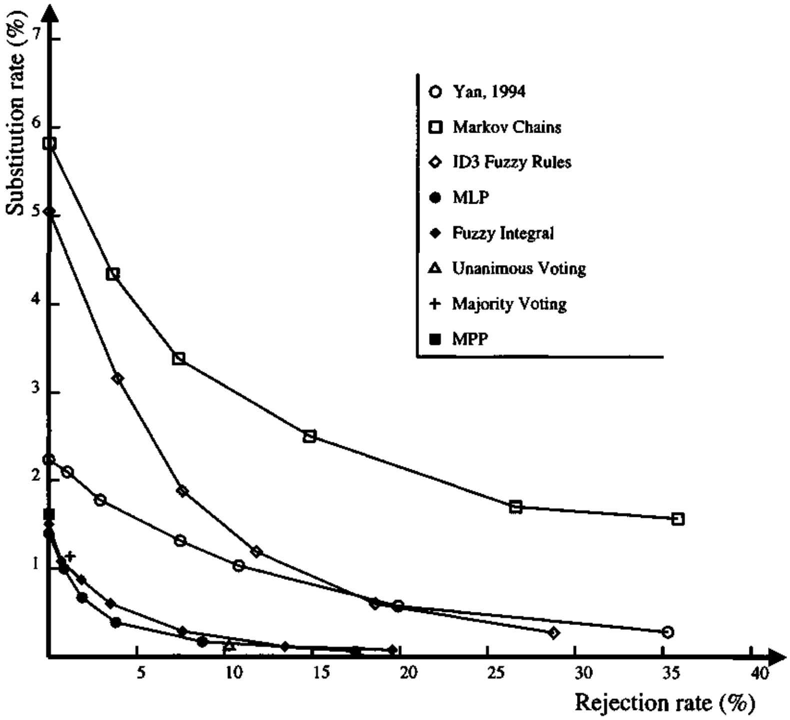


Figure 7.7 Substitution/rejection rates for various combination classifiers (MLP: multilayer perceptron; MPP: maximum posteriori probability).

Bibliography

[1] L. A. Zadeh. Fuzzy sets. *Information and Controls*, 8:338–353, 1965.

[2] J. C. Bezdek and S. K. Pal. *Fuzzy Models for Pattern Recognition.* IEEE Press, New York, 1992.

[3] B. R. Gaines and L. J. Kohout. The fuzzy decade: A bibliography of fuzzy systems and closely related topics. *Int. J. Man-Machine Studies*, 9:1–68, 1977.

[4] R. R. Yager, S. Ovchinnikov, R. M. Tong, and H. T. Nguyen, editors. *Fuzzy Sets and Applications: Selected Papers by L. A. Zadeh.* John Wiley & Sons, New York, 1987.

[5] H. J. Zimmermann. *Fuzzy Set Theory and Its Applications.* Kluwer, Boston, 1991.

[6] G. J. Klir and T. A. Folger. *Fuzzy Sets, Uncertainty, and Information.* Prentice Hall, Englewood, 1988.

[7] A. Kaufmann and M. M. Gupta. *Fuzzy Mathematical Models in Engineering and Management Science.* Elsevier, Amsterdam, 1988.

[8] L. A. Zadeh. A fuzzy-algorithmic approach to the definition of complex or imprecise concepts. *Int. J. Man-Machine Studies*, 8:249–291, 1976.

[9] L. A. Zadeh. The concept of a linguistic variable and its application to approximate reasoning, Part III. *Information Sciences*, 8:301–357, 1975.

[10] L. A. Zadeh. Fuzzy logic. *IEEE Computer*, pages 83–93, April 1988.

[11] H. Ogawa. Labeled point pattern matching by fuzzy relaxation. *Pattern Recognition*, 17(5):569–573, 1984.

[12] H. Ogawa. A fuzzy relaxation technique for partial shape matching. *Pattern Recognition Letters*, 15(4):349–355, 1993.

[13] C. C. Lee. Fuzzy logic in control systems: fuzzy logic controller — Part I. *IEEE Transactions on Systems, Man, and Cybernetics*, 20(2):404–418, 1990.

[14] C. C. Lee. Fuzzy logic in control systems: fuzzy logic controller — Part II. *IEEE Transactions on Systems, Man, and Cybernetics*, 20(2):419–435, 1990.

[15] R. Krishapuram and J. Lee. Fuzzy-connective-based hierarchical aggregation networks for decision making. *Fuzzy Sets and Systems*, (46):11–27, 1992.

[16] A. De Luca and S. Termini. A definition of a nonprobabilistic entropy in the setting of fuzzy sets theory. *Information and Control*, 20:301–312, 1972.

[17] S. K. Pal and R. A. King. Image enhancement using smoothing with fuzzy sets. *IEEE Transactions on Systems, Man, and Cybernetics*, 11:494–501, 1981.

[18] A. O. Esogbue and R. C. Elder. Fuzzy sets and the modeling of physician decision processes: Part I: the initial interview-information gathering process. *Fuzzy Sets and Systems*, 2:279–291, 1979.

[19] A. O. Esogbue and R. C. Elder. Fuzzy sets and the modeling of physician decision processes: Part II: fuzzy diagnosis decision models. *Fuzzy Sets and Systems*, 3:1–9, 1980.

[20] M. Sugeno. Fuzzy measures and fuzzy integrals — A survey. In M. M. Gupta, G. N. Saridis, and B. R. Gaines, editors, *Fuzzy Automata and Decision Processes*, pages 89–102. North-Holland Publishing Company, Amsterdam, Netherlands, 1977.

[21] S. B. Cho and J. H. Kim. Combining multiple neural networks by fuzzy integral for robust classification. *IEEE Transactions on Systems, Man, and Cybernetics*, 25:380–384, 1995.

[22] H. T. Nguyen M. Grabish and E. A. Walker. *Fundamentals of Uncertainty Calculi with Application to Fuzzy Inference.* Kluwer, Dordrecht (Netherlands), 1995.

[23] J. Dickerson and B. Kosko. Fuzzy function learning with covariance ellipsoids. In *Proceedings of IEEE International Conference on Neural Networks*, volume III, pages 1162–1167, San Francisco, U.S.A., 1993.

[24] Z. Chi and H. Yan. Image segmentation using fuzzy rules derived from K-means clusters. *Journal of Electronic Imaging*, 4(2):199–206, 1995.

[25] Z. Chi and H. Yan. AVQ based fuzzy rules for image segmentation. In *Proceedings of the Fifth Australian Conference on Neural Networks (ACNN'94)*, pages 137–140, Brisbane, Australia, February 1994.

[26] Z. Chi, J. Wu, and H. Yan. Handwritten numeral recognition using self-organizing maps and fuzzy rules. *Pattern Recognition*, 28(1):59–66, 1995.

[27] Z. Chi and H. Yan. Handwritten numeral recognition using a small number of fuzzy rules with optimized defuzzification parameters. *Neural Networks*, 8(5):821–827, 1995.

[28] D. Driankov, H. Hellendoorn, and M. Reinfrank. *An Introduction to Fuzzy Control.* Springer-Verlag, 1993.

[29] C. T. Sun and J. S. Jang. A neuro-fuzzy classifier and its applications. In *Proceedings of Second IEEE International Conference on Fuzzy Systems*, pages 94–98, San Francisco, U.S.A., 1993.

[30] T. Kohonen. The self-organizing map. *Proceedings of IEEE*, 78(9):1464–1480, 1990.

[31] S. U. Lee and S. Y. Chung. A comparative performance study of several global thresholding techniques for segmentation. *Computer Vision, Graphics and Image Processing*, 52:171–190, 1990.

[32] P. K. Sahoo, S. Soltani, and A. K. C. Wong. A survey of thresholding techniques. *Computer Vision, Graphics and Image Processing*, 41:233–260, 1988.

[33] R. C. Gonzalez and R. E. Woods. *Digital Image Processing.* Addison-Wesley Publishing Company, Reading, MA, 1992.

[34] C. W. Therrien. *Decision Estimation and Classification, An Introduction to Pattern Recognition and Related Topics.* John Wiley & Sons, New York, 1989.

[35] S. K. Pal and A. Rosenfeld. Image enhancement and thresholding by optimization of fuzzy compactness. *Pattern Recognition Letters*, 7:77–86, 1988.

[36] S. K. Pal and A. Ghosh. Fuzzy geometry in image analysis. *Fuzzy Sets and Systems*, 48:23–40, 1992.

[37] L. K. Huang and M. J. Wang. Image thresholding by minimizing the measure of fuzziness. *Pattern Recognition*, 28:41–51, 1995.

[38] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979.

[39] J. Kittler and J. Illingworth. Minimum error thresholding. *Pattern Recognition*, 19:41–47, 1986.

[40] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C, the Art of Scientific Computing.* Cambridge University Press, Cambridge, 1988.

[41] H. Li and H. S. Yang. Fast and reliable image enhancement using fuzzy relaxation technique. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(5):1276–1281, 1989.

[42] H. Wang and H. Yan. Text extraction from color map images. *J. Electronic Imaging*, 3:390–396, 1994.

[43] J. Wu, H. Yan, and A. Chalmers. Color image segmentation using fuzzy clustering and supervised learning. *J. Electronic Imaging*, 3:397–403, 1994.

[44] H. Yan and J. Wu. Character and line extraction from color map images using a multilayer neural network. *Pattern Recognition Letters*, 15:97–103, 1994.

[45] J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, New York, 1981.

[46] M. P. Windham. Cluster validity for the fuzzy c-means clustering algorithm. *IEEE Trans. Pattern Anal. Machine Intell*, 4(4):357–363, 1982.

[47] X. L. Xie and G. Beni. A validity measure for fuzzy clustering. *IEEE Trans. Pattern Anal. Machine Intell*, 13(8):841–847, 1991.

[48] S. M. Cargill, R. F. Meyer, D. D. Picklyk, and F. Urquidi. Summary of resource assessment methods resulting from the international geological correlation program project 98. *Math. Geol.*, 9:211–220, 1977.

[49] M. David. *Geostatistical Ore Reserve Estimation*. Elsevier Scientific Publishing Company, Amsterdam, 1977.

[50] S. N. Carras. Concepts for estimating ore reserves and a comparative view of resource estimation methods. In *Resources and Reserves Symp.*, pages 73–80, AusIMM Sydney Branch, Australia, 1987.

[51] C. J. Mann. Toward a theoretical stratigraphy. *Math. Geol.*, 9:649–652, 1977.

[52] S. V. L. N. Rao and J. Prasad. Definition of kriging in terms of fuzzy logic. *Math. Geol.*, 14:37–42, 1982.

[53] G. Granath. Application of fuzzy clustering and fuzzy classification to evaluate the provenance of glacial till. *Math. Geol.*, 16:283–300, 1984.

[54] A. Bardossy, I. Bogardi, and W. E. Kelly. Kriging with imprecise (fuzzy) variograms. II: Application. *Math. Geol.*, 22:81–94, 1990.

[55] A. Bardossy, I. Bogardi, and W. E. Kelly. Kriging with imprecise (fuzzy) variograms. I: Theory. *Math. Geol.*, 22:63–79, 1990.

[56] R. R. Yager and D. P. Filev. *Essentials of Fuzzy Modeling and Control.* John Wiley & Sons, Inc., 1994.

[57] I. Clark. *Practical Geostatistics.* Applied Science Publishers, London, 1979.

[58] L. H. Chen and J. R. Lieh. Handwritten character recognition using a 2-layer random graph model by relaxation matching. *Pattern Recognition*, 23(11):1189–1205, 1990.

[59] F. H. Cheng, W. H. Hsu, and C. A. Chen. Fuzzy approach to solve the recognition problem of handwritten Chinese characters. *Pattern Recognition*, 22(2):133–141, 1989.

[60] D. G. Sim, Y. K. Ham, and R. H. Park. On-line recognition of cursive Korean characters using DP matching and fuzzy concept. *Pattern Recognition*, 27(12):1605–1620, 1994.

[61] J. R. Quinlan. Induction of decision trees. *Machine Learning*, (1):81–106, 1986.

[62] H. Takagi and I. Hayashi. NN-driven fuzzy reasoning. *International Journal of Approximate Reasoning*, 5(3):191–212, 1991.

[63] T. Furuya, A. Kokuba, and T. Sakamoto. NFS: neuro fuzzy inference system. In *Proc. Iizuka-88*, pages 219–230, Iizuka, Japan, 1988.

[64] K. Saitoh and R. Nakano. Medical diagnostic expert system based on PDP model. In *Proc. ICNN'88*, pages 255–262, 1988.

[65] M. M. Gupta, W. Pedrycz, and J. Kiszka. Fuzzy control: from fuzzy controllers to cognitive controllers. In *Proc. 3rd IFSA Congress*, pages 258–261, Seattle, Wash., 1989.

[66] J. M. Keller, R. Krishnapuram, and F. C. H. Rhee. Evidence aggregation networks for fuzzy logic inference. *IEEE Transactions on Neural Networks*, 3(5):761–769, 1992.

[67] T. Yamakawa and S. Tomoda. A fuzzy neuron and its application to pattern recognition. In *Proc. 3rd IFSA Congress*, pages 30–38, Seattle, Wash., 1989.

[68] J. Keller and H. Tahani. Backpropagation neural networks for fuzzy logic. *Information Science*, 1992.

[69] I. B. Turksen, L. Guo, C. Lucas, and K. Smith. Hardware realization of fuzzy logic and artificial neural networks. In *Proc. Int. Conf. on Fuzzy Logic and Neural Networks*, pages 133–136, Iizuka, Japan, 1990.

[70] J. M. Keller and H. Tahani. Implementation of conjunctive and disjunctive fuzzy logic rules with neural networks. *International Journal of Approximate Reasoning*, 6:221–240, 1992.

[71] R. Krishapuram. Fuzzy-set-based hierarchical networks for information fusion in computer vision. *Neural Networks*, 5:335–350, 1992.

[72] R. Krishnapuram and F. Chung-Hoon Rhee. Compact fuzzy rule base generation methods for computer vision. In *Proceedings of Second IEEE International Conference on Fuzzy Systems*, pages 809–814, San Francisco, U.S.A., 1993.

[73] C. Hung and B. R. Fernandez. Minimizing rules of fuzzy logic system by using a systematic approach. In *Proceedings of Second IEEE International Conference on Fuzzy Systems*, pages 38–43, San Francisco, U.S.A., 1993.

[74] L. Wang and J. M. Mendel. Generating fuzzy rules by learning from examples. In *Proceedings of the 1991 IEEE International Symposium on Intelligent Control*, pages 263–268, Arlington, Virginia, U.S.A., 1991.

[75] Z. Chi and H. Yan. Map image segmentation based on thresholding and fuzzy rules. *Electronics Letters*, 29(21):1841–1843, 1993.

[76] J. R. Quinlan. Simplifying decision trees. *Int. J. Man-Machine Studies*, 27:221–234, 1987.

[77] P. E. Maher and D. St. Clair. Uncertain reasoning in an ID3 machine learning framework. In *Proceedings of Second IEEE International Conference on Fuzzy Systems*, pages 7–12, San Francisco, U.S.A., 1993.

[78] Z. Chi and H. Yan. ID3-derived fuzzy rules and optimized defuzzification for handwritten numeral recognition. *IEEE Transactions on Fuzzy Systems*, 4(1):24–31, 1996.

[79] L. Fu. Rule generation from neural networks. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(8):1114–1124, 1994.

[80] A. K. Chhabra, Z. An, D. Balick, G. Cerf, K. Loris, P. Sheppard, R. Smith, and B. Witter. High-order statistically derived combinations of geometric features for handprinted character recognition. In *2nd Int. Conf. on Document Analysis and Recognition*, pages 397–401, 1993.

[81] Z. Chi and H. Yan. Feature evaluation and selection based on an entropy measurement with data clustering. *Optical Engineering*, 34(12):3514–3519, 1995.

[82] P. Siy and C. S. Chen. Fuzzy logic for handwritten numeral character recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, pages 570–574, 1974.

[83] M. D. Garris and R. A. Wilkinson. NIST special database 3 handwritten segmented characters, 1992. National Institute of Standards and Technology (NIST).

[84] L. Xu, A. Krzyzak, and C. Suen. Methods of combining multiple classifiers and their applications to handwritten recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(3):418–435, 1992.

[85] C. Nadal, R. Legault, and C. Suen. Complementary algorithms for the recognition of unconstrained handwritten numerals. In *Proceedings of the Tenth International Conference on Pattern Recognition*, pages 443–449, 1990.

[86] R. Battiti and A. Colla. Democarcy in neural nets: voting schemes for classification. *Neural Networks*, 7(4):691–707, 1994.

[87] E. Mandler and J. Schurmann. Combining the classification results of independent classifiers based on the Dempster/Shaft theory of evidence. In E. Gelsema and L. Kanal, editors, *Pattern Recognition and Artificial Intelligence*, pages 381–393. Elsevier Science Publishers B. V., 1988.

[88] G. Rogova. Combining the results of several neural network classifiers. *Neural Networks*, 7(5):777–781, 1994.

[89] L. Xu, A. Krzyzak, and C. Suen. Associative switch for combining multiple classifiers. *Journal of Artificial Neural Networks*, 1(1):77–100, 1994.

[90] Z Kovacs-V., R. Guerrieri, and G. Baccarani. Cooperative classifiers for high quality hand-printed character recognition. In *World Congress on Neural Networks*, pages I-186–I-189, 1993.

[91] M. Sugeno. Fuzzy measures and fuzzy integrals — a survey. In M. M. Gupta, G. N. Saridis, and B. R. Gaines, editors, *Fuzzy Automata and Decision Processes*, pages 89–102. North-Holland Publishing Company, Amsterdam, Netherlands, 1977.

[92] H. Tahani and J. M. Keller. Information fusion in computer vision using the fuzzy integral. *IEEE Transactions on Systems, Man, and Cybernetics*, 20:733–741, 1990.

[93] Z. Chi, M. Suters, and H. Yan. Handwritten numeral recognition using combined ID3-derived fuzzy rules and Markov chains. *Pattern Recognition* (paper accepted).

[94] S. T. Wierzchon. On fuzzy measure and fuzzy integral. In M. M. Gupta and E. Sanchez, editors, *Fuzzy Information and Decision Processes*, pages 79–86. North-Holland Publishing Company, Amsterdam, Netherlands, 1982.

[95] H. Yan. Handwritten digit recognition using optimized prototypes. *Pattern Recognition Letters*, 15:207–211, 1994.

[96] B. V. Dasarathy, editor. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques.* IEEE Computer Society Press, Los Almitos, CA, 1991.

[97] H. Yan. Prototype optimization for nearest neighbor classifiers using a two-layer neural network. *Pattern Recognition*, 26:317–324, 1993.

Index

α, 5
g_λ-fuzzy measure, 194
n-ary fuzzy relation, 12

actual data distribution, 17
adaptive thresholding, 168
adaptive vector quantization (AVQ), 26
algebraic form, 156
algebraic simplification of switching functions, 156
angle difference, 23
asymmetric membership functions, 20

backpropagation algorithm, 192
basic line pattern matching algorithm, 117
batch mode, 162
Bayes decision rule, 47
bell-shaped membership function, 18, 39
bounded product, 11
bounded sum, 11

c-means algorithm, 86
c-means clustering, 24
c-means clustering procedure, 87
c-means clusters, 165, 174
centroid defuzzification, 35, 161, 167
chain generation, 202
character database, 174, 178
characteristic function, 3
choosing contact lenses, 147
Choquet integral, 195
class labels, 139
class separability, 54
classification problems, 142
closest cluster center, 87
cluster centers, 24, 86
cluster validity, 91
cluster variance, 24
clustering, 23
clustering process, 86
color image segmentation, 95
combined classifiers, 189, 204, 210
compactness and separation validity function, 93
comparison between hard and fuzzy clustering algorithms, 89
comparison of thresholding methods, 84
compensative connectives, 152
completely-specified rules, 157
Compositional rule, 13
conflicting rules, 142
confusion matrix, 190
conjugate gradient algorithm, 54
Conjunctive rule, 12
continuity of fuzzy set, 194
convex, 5
convex and concave curvatures, 174
convex combination, 8
corner points, 179
correct classification rate, 190
correction factors, 196
coupled control points, 103
covariance ellipsoid, 28
crisp sets, 3, 144
crossover point, 5
crosspoint level, 21, 33
crosspoints, 21
curved segments, 178

decision regions, 48

decision tree approach for generating fuzzy rules, 145
decision trees, 145, 183
defuzzification, 151, 159
defuzzification method, 145
defuzzification scheme, 103
defuzzification using neural networks, 183
defuzzification using two-layer perceptron, 161
degree of overlapping, 26, 33
degree of truth, 3
design of a fuzzy rule-based recognition system, 187
difference intensity (DI), 164
direction cosines, 28
discriminant analysis, 54
disemblance measure, 23
document image segmentation, 89
domain intervals, 142

eigenvalues, 28
Entailment rule, 12
entropy measure, 59, 117
error function, 36, 42, 162, 192
estimation of resources, 100
Euclidean distance, 23, 86, 88
Euclidean distance measure, 24, 27, 208
Euclidean half-length, 28
evenly-distributed membership functions, 21
extra lines in a line pattern, 125

FCM clustering procedure, 88
feature map, 28
feature selection, 174, 202
fired fuzzy rules, 35
firing degree, 35
first-order estimation, 26
first-order learning, 192
Fisher criterion, 55, 60
fuzzified decision rules, 150
fuzziness, 2, 26, 33, 100
fuzzy aggregation networks, 152
fuzzy binary relation, 12
fuzzy c-means, 100
fuzzy c-means algorithm, 87
fuzzy clustering, 14
fuzzy compactness measure, 46
fuzzy complement, 8
fuzzy concentration, 10
fuzzy densities, 195, 197
fuzzy dilation, 10
fuzzy entropy, 15, 46
fuzzy grade matrix, 151, 182
fuzzy inference system, 100
fuzzy integral, 15
fuzzy integral approach, 208
fuzzy integral model for classifier fusion, 195
fuzzy integrals, 195
fuzzy intensification, 11
fuzzy interpolation, fuzzy extrapolation, 102
fuzzy intersection, 8
fuzzy logic, 3
fuzzy measure, 15, 194
fuzzy membership function, 14
fuzzy minus, 10
fuzzy models for image processing and pattern recognitions, 14
fuzzy parameter estimation, 100
fuzzy patches, 26
fuzzy pattern matching, 14
fuzzy plus, 10
fuzzy relations, 11
fuzzy relaxation, 112
fuzzy rule minimization using data clustering, 156
fuzzy rule minimization using Karnaugh Maps, 156
fuzzy rule-based systems, 14, 139
fuzzy rules, 139, 159, 165, 178
fuzzy search domain, 100
fuzzy set, 3
fuzzy sets, 3
fuzzy singleton, 5

fuzzy space-grade system, 100
fuzzy subsets, 18, 156
fuzzy system, 17
fuzzy thresholding algorithm, 57
fuzzy union, 8
fuzzying the input space, 142

Gaussian distributions, 50
Gaussian probability densities, 52
Gaussian-type function, 102
generalized confusion matrix, 190
generalized mean operator, 152
Generalized modus ponens, 13
grades of cluster centers, 102
gradient descent algorithm, 35, 162
gradient information for numerical optimization, 53

half-height points, 39
handwritten Chinese character recognition, 112, 134
handwritten numeral recognition, 178
hard partition, 142
harmonic mean, 152
heuristic selections, 18
Huang and Wang's fuzzy thresholding algorithm, 46
Huang and Wang's thresholding method, 57
hyper-ellipsoids, 24, 28

ID3-derived fuzzy rule-based classifier, 178, 186
ID3-derived fuzzy rules, 197
IF-THEN rules, 139
image pixel intensity histogram, 52
image processing, 14
image thresholding, 47
incompletely-specified rules, 157, 158
index of area coverage, 46
index of fuzziness, 46
index of nonfuzziness, 46
information-based method, 146
intersection, 152
intersection connectives, 152
inverse square distance, 106
iron-ore deposit, 104

junction points, 179

Kittler and Illingworth's minimum error thresholding method, 46
kriging, 106

labeled data, 85
Law of Excluded Middle, 3
learning coefficient, 26
learning rate, 37
left width of a membership function, 20
level-cut, 5
line convolutions, 174
line location similarity, 112
line orientation similarity, 113
line pattern matching, 14, 109
line relation similarity, 114
linguistic fuzzy model, 102
linguistic rules, 139
linguistic terms, 18, 139
local centroids, 26
local contrast (J), 164
local covariance matrix, 27

Machine Learning Databases, 147
majority voting, 191, 206
map segmentation, 139
Markov chain, 202
Markov chain-based classifier, 201
matching cost, 119
matching degree, 42, 160
maximum accumulated matching defuzzification, 160, 178
maximum matching defuzzification, 160
maximum posteriori probability, 206
maxterm expression, 156
membership function merging, 33
membership function tuning, 35
membership functions, 3, 17, 26, 28, 33, 57, 102, 112, 113, 117, 142, 151, 152, 165, 174, 182, 197

minimization of fuzzy rules, 153
minimum error thresholding method, 56
minimum product-of-sums rules, 156
minimum sum-of-products, 157
minimum sum-of-products rules, 156
minterm expression, 156
missing lines in a line pattern, 125
model fitting method for threshold selection, 52
monotonicity of fuzzy set, 194
MR image segmentation, 93
multi-classifier combination for handwritten numeral classification, 197
multi-classifier combination using a fuzzy integral model, 195
multi-classifier combination using a multilayer perceptron, 192
multi-classifier combination using majority voting, 191
multi-classifier combination using unanimous voting, 191
multi-classifier using maximum posteriori probability (MPP), 191
multi-valued logic, 3
multilevel thresholding, 63

nearest neighbor classifier, 198
neural network approach of tuning membership functions, 38
NIST Special Database 3, 178, 180
noisy data, 150
noisy line patterns, 125
non-circular branch, 179
non-convex, 5
non-fuzzy outputs, 35
non-iterative thresholding algorithms, 61
normal, 5
numerical optimization, 53

optimal threshold, 52
optimized defuzzification, 159, 186
optimized defuzzification parameters, 161
optimized nearest neighbor classifier, 198
optimized prototypes, 199
Otsu's thresholding method, 46, 54

parameter identification, 104
partial shape matching, 112
partition coefficient, 91
partition entropy, 91
pattern centroids, 27
pattern recognition, 2, 14
peak point, 20
point pattern matching, 109, 134
posterior probability, 47
printed upper-case English letter recognition, 174
priori probabilities, 54, 66
probability, 1
probability of error, 48
Projection rule, 13

quantization vectors, 26

randomness and fuzziness, 192
recursive method, 60
rejection rate, 190
resemblance measure, 23
right width of a membership function, 20
rotated line patterns, 129
rule bank, 144
rule degree, 142
rule generation using neural networks, 152
rules for fuzzy inference, 12

S-function, 11, 18
search-distance radius, 100
second-order estimation, 27
second-order learning, 192
segmentation of building images, 70
segmentation of document images, 69
segmentation of human images, 71
segmentation of map images using fuzzy rules, 163
self-organizing map (SOM), 28

Shannon's entropy function, 59
sigmoidal transfer function, 42, 43, 163
sign operator, 38, 164
significance level test, 150
similarity measures, 23
simplified decision rules, 150, 183
Siy and Chen's fuzzy line labeling, 179
SOM prototypes, 31
soundness degree, 142
standard deviation (SD), 164
steepest descent algorithm, 37
straightness of a line, 179
stroke order, 110
subjective membership functions, 17
subnormal, 5
substitution rate, 190
Sugeno integral, 195
supervised c-means clustering algorithm, 178
supervised pattern recognition method, 85
symmetric membership functions, 20
synaptic connection matrix, 26
synaptic vectors, 27

three classifier types, 189
thresholding, 45
tips (end points), 179
topology-preserving property, 31
trapezoidal membership function, 18
triangular membership function, 18
tuning procedure, 38
twelve types of segments, 179

unanimous voting, 191, 206
uncertain data, 150
unified formulation of thresholding algorithms, 61
union, 152
union connectives, 152
unit eigenvectors, 28
unlabeled data, 85
unsupervised competitive learning, 26
unsupervised pattern recognition method, 85

vagueness, 3
voting schemes, 190

Wang and Mendel's fuzzy rule generation technique, 141
width of a membership function, 39

Yan's optimized nearest neighbor classifier, 199