1995

Fuzzy inference networks for pattern recognition.

Yaling Cai

University of Windsor
NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

Canada
Fuzzy Inference Networks

for Pattern Recognition

by

Yaling Cai

A Dissertation
Submitted to the Faculty of Graduate Studies and Research
Through the Department of Electrical Engineering in
Partial Fulfilment of the Requirements for the
Degree of Doctor of Philosophy at the
University of Windsor

Windsor, Ontario, Canada

1995

(c) 1995 Yaling Cai
THE AUTHOR HAS GRANTED AN IRREVOCABLE NON-EXCLUSIVE LICENCE ALLOWING THE NATIONAL LIBRARY OF CANADA TO REPRODUCE, LOAN, DISTRIBUTE OR SELL COPIES OF HIS/HER THESIS BY ANY MEANS AND IN ANY FORM OR FORMAT, MAKING THIS THESIS AVAILABLE TO INTERESTED PERSONS.

L'AUTEUR A ACCORDE UNE LICENCE IRREVOCABLE ET NON EXCLUSIVE PERMETTANT À LA BIBLIOTHÈQUE NATIONALE DU CANADA DE REPRODUIRE, PRETER, DISTRIBUER OU VENDRE DES COPIES DE SA THÈSE DE QUELQUE MANIÈRE ET SOUS QUELQUE FORME QUE CE SOIT POUR METTRE DES EXEMPLAIRES DE CETTE THÈSE À LA DISPOSITION DES PERSONNE INTERESSÉES.

THE AUTHOR RETAINS OWNERSHIP OF THE COPYRIGHT IN HIS/HER THESIS. NEITHER THE THESIS NOR SUBSTANTIAL EXTRACTS FROM IT MAY BE PRINTED OR OTHERWISE REPRODUCED WITHOUT HIS/HER PERMISSION.

L'AUTEUR CONSERVE LA PROPRIETE DU DROIT D'AUTEUR QUI PROTEGE SA THÈSE. NI LA THÈSE NI DES EXTRAITS SUBSTANTIELS DE CELLE-CI NE DOIVENT ÊTRE IMPRIMÉS OU AUTREMENT REPRODUITS SANS SON AUTORISATION.

ISBN 0-612-01436-3
Subject Categories

**THE HUMANITIES AND SOCIAL SCIENCES**

<table>
<thead>
<tr>
<th>COMMUNICATIONS AND THE ARTS</th>
<th>0729</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture</td>
<td>0729</td>
</tr>
<tr>
<td>Art History</td>
<td>0737</td>
</tr>
<tr>
<td>Cinema</td>
<td>0716</td>
</tr>
<tr>
<td>Dance</td>
<td>0726</td>
</tr>
<tr>
<td>Fine Arts</td>
<td>0727</td>
</tr>
<tr>
<td>Information Science</td>
<td>0723</td>
</tr>
<tr>
<td>Journalism</td>
<td>0739</td>
</tr>
<tr>
<td>Library Science</td>
<td>0708</td>
</tr>
<tr>
<td>Mass Communications</td>
<td>0708</td>
</tr>
<tr>
<td>Speech Communication</td>
<td>0413</td>
</tr>
<tr>
<td>Theater</td>
<td>0456</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EDUCATION</th>
<th>0515</th>
</tr>
</thead>
<tbody>
<tr>
<td>General</td>
<td>0515</td>
</tr>
<tr>
<td>Administration</td>
<td>0514</td>
</tr>
<tr>
<td>Adult Continuing</td>
<td>0516</td>
</tr>
<tr>
<td>Agricultural</td>
<td>0517</td>
</tr>
<tr>
<td>Art</td>
<td>0723</td>
</tr>
<tr>
<td>Bilingual and Multicultural</td>
<td>0724</td>
</tr>
<tr>
<td>Business</td>
<td>0518</td>
</tr>
<tr>
<td>Community College</td>
<td>0519</td>
</tr>
<tr>
<td>Community Institutions</td>
<td>0520</td>
</tr>
<tr>
<td>Early Childhood</td>
<td>0521</td>
</tr>
<tr>
<td>Elementary</td>
<td>0522</td>
</tr>
<tr>
<td>Finance</td>
<td>0577</td>
</tr>
<tr>
<td>Guidance and Counseling</td>
<td>0519</td>
</tr>
<tr>
<td>Health</td>
<td>0580</td>
</tr>
<tr>
<td>Higher Education</td>
<td>0545</td>
</tr>
<tr>
<td>History</td>
<td>0520</td>
</tr>
<tr>
<td>Human Geomatics</td>
<td>0521</td>
</tr>
<tr>
<td>Industrial</td>
<td>0522</td>
</tr>
<tr>
<td>Language and Literature</td>
<td>0523</td>
</tr>
<tr>
<td>Multicultural</td>
<td>0523</td>
</tr>
<tr>
<td>Music</td>
<td>0523</td>
</tr>
<tr>
<td>Physical Philosophy</td>
<td>0523</td>
</tr>
<tr>
<td>Psychology</td>
<td>0523</td>
</tr>
<tr>
<td>Reading</td>
<td>0535</td>
</tr>
<tr>
<td>Religious</td>
<td>0533</td>
</tr>
<tr>
<td>Religious Science</td>
<td>0714</td>
</tr>
<tr>
<td>Secondary</td>
<td>0533</td>
</tr>
<tr>
<td>Social Sciences</td>
<td>0533</td>
</tr>
<tr>
<td>Sociology</td>
<td>0540</td>
</tr>
<tr>
<td>Special</td>
<td>0529</td>
</tr>
<tr>
<td>Teacher Training</td>
<td>0530</td>
</tr>
<tr>
<td>Technology</td>
<td>0710</td>
</tr>
<tr>
<td>Tests and Measurements</td>
<td>0288</td>
</tr>
<tr>
<td>Vocational</td>
<td>0747</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LANGUAGE LITERATURE AND LINGUISTICS</th>
<th>0579</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language</td>
<td>0579</td>
</tr>
<tr>
<td>General</td>
<td>0579</td>
</tr>
<tr>
<td>Classical</td>
<td>0294</td>
</tr>
<tr>
<td>Comparative</td>
<td>0295</td>
</tr>
<tr>
<td>Medieval</td>
<td>0527</td>
</tr>
<tr>
<td>Modern</td>
<td>0521</td>
</tr>
<tr>
<td>African</td>
<td>0316</td>
</tr>
<tr>
<td>American</td>
<td>0591</td>
</tr>
<tr>
<td>Asian</td>
<td>0325</td>
</tr>
<tr>
<td>Canadian (English)</td>
<td>0325</td>
</tr>
<tr>
<td>Canadian (French)</td>
<td>0335</td>
</tr>
<tr>
<td>Celtic</td>
<td>0345</td>
</tr>
<tr>
<td>German</td>
<td>0311</td>
</tr>
<tr>
<td>Latin American</td>
<td>0312</td>
</tr>
<tr>
<td>Middle Eastern</td>
<td>0334</td>
</tr>
<tr>
<td>Romance</td>
<td>0313</td>
</tr>
<tr>
<td>Slavic and East European</td>
<td>0314</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PHILOSOPHY, RELIGION AND THEOLOGY</th>
<th>0522</th>
</tr>
</thead>
<tbody>
<tr>
<td>Philosophy</td>
<td>0522</td>
</tr>
<tr>
<td>General</td>
<td>0518</td>
</tr>
<tr>
<td>Biblical Studies</td>
<td>0211</td>
</tr>
<tr>
<td>Clergy</td>
<td>0319</td>
</tr>
<tr>
<td>History of Philosophy</td>
<td>0522</td>
</tr>
<tr>
<td>Philosophy of Religion</td>
<td>0522</td>
</tr>
<tr>
<td>Theology</td>
<td>0449</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SOCIAL SCIENCES</th>
<th>0323</th>
</tr>
</thead>
<tbody>
<tr>
<td>American Studies</td>
<td>0323</td>
</tr>
<tr>
<td>Anthropology</td>
<td>0324</td>
</tr>
<tr>
<td>Art</td>
<td>0326</td>
</tr>
<tr>
<td>Physical</td>
<td>0327</td>
</tr>
<tr>
<td>Business Administration</td>
<td>0310</td>
</tr>
<tr>
<td>General</td>
<td>0310</td>
</tr>
<tr>
<td>Accounting</td>
<td>0453</td>
</tr>
<tr>
<td>Banking</td>
<td>0770</td>
</tr>
<tr>
<td>Management</td>
<td>0549</td>
</tr>
<tr>
<td>Marketing</td>
<td>0358</td>
</tr>
<tr>
<td>Canadian Studies</td>
<td>0385</td>
</tr>
<tr>
<td>Economics</td>
<td>0501</td>
</tr>
<tr>
<td>General</td>
<td>0501</td>
</tr>
<tr>
<td>Agricultural</td>
<td>0503</td>
</tr>
<tr>
<td>Commerce-Business</td>
<td>0503</td>
</tr>
<tr>
<td>Finance</td>
<td>0533</td>
</tr>
<tr>
<td>History</td>
<td>0509</td>
</tr>
<tr>
<td>Labor</td>
<td>0510</td>
</tr>
<tr>
<td>Laboral</td>
<td>0511</td>
</tr>
<tr>
<td>Folklore</td>
<td>0530</td>
</tr>
<tr>
<td>Geography</td>
<td>0369</td>
</tr>
<tr>
<td>Genealogy</td>
<td>0369</td>
</tr>
<tr>
<td>History</td>
<td>0507</td>
</tr>
<tr>
<td>General</td>
<td>0578</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>THE SCIENCES AND ENGINEERING</th>
<th>0578</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIOLOGICAL SCIENCES</td>
<td>0578</td>
</tr>
<tr>
<td>Agriculture</td>
<td>0733</td>
</tr>
<tr>
<td>Animal Culture</td>
<td>0475</td>
</tr>
<tr>
<td>Animal Nutrition</td>
<td>0476</td>
</tr>
<tr>
<td>Animal Pathology</td>
<td>0719</td>
</tr>
<tr>
<td>Animal Physiology</td>
<td>0717</td>
</tr>
<tr>
<td>Biology</td>
<td>0506</td>
</tr>
<tr>
<td>Anatomy</td>
<td>0506</td>
</tr>
<tr>
<td>Biostatistics</td>
<td>0308</td>
</tr>
<tr>
<td>Botany</td>
<td>0259</td>
</tr>
<tr>
<td>Cell Biology</td>
<td>0259</td>
</tr>
<tr>
<td>Dermatology</td>
<td>0253</td>
</tr>
<tr>
<td>Endocrinology</td>
<td>0369</td>
</tr>
<tr>
<td>Entomology</td>
<td>0793</td>
</tr>
<tr>
<td>Entomology</td>
<td>0793</td>
</tr>
<tr>
<td>Microbiology</td>
<td>0310</td>
</tr>
<tr>
<td>Molecular Biology</td>
<td>0317</td>
</tr>
<tr>
<td>Neuroscience</td>
<td>0414</td>
</tr>
<tr>
<td>Physiology</td>
<td>0433</td>
</tr>
<tr>
<td>Radiation</td>
<td>0821</td>
</tr>
<tr>
<td>Zoology</td>
<td>0472</td>
</tr>
<tr>
<td>Biophysics</td>
<td>0786</td>
</tr>
<tr>
<td>Medical</td>
<td>0786</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EARTH SCIENCES</th>
<th>0425</th>
</tr>
</thead>
<tbody>
<tr>
<td>Biogeochemistry</td>
<td>0425</td>
</tr>
<tr>
<td>Geochemistry</td>
<td>0976</td>
</tr>
</tbody>
</table>

| GEOLGY                           | 0370 |
| Geology                          | 0370 |
| Geophysics                       | 0386 |
| Hydrology                        | 0386 |
| Mineralogy                       | 0411 |
| Paleobotany                      | 0466 |
| Paleocology                      | 0476 |
| Paleontology                     | 0476 |
| Paleontology                     | 0476 |
| Physical Geography               | 0367 |
| Physical Geochemistry            | 0315 |

<table>
<thead>
<tr>
<th>HEALTH AND ENVIRONMENTAL SCIENCES</th>
<th>0748</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environmental Sciences</td>
<td>0748</td>
</tr>
<tr>
<td>Health Sciences</td>
<td>0546</td>
</tr>
<tr>
<td>General</td>
<td>0546</td>
</tr>
<tr>
<td>Audiology</td>
<td>0300</td>
</tr>
<tr>
<td>Chemotherapy</td>
<td>0972</td>
</tr>
<tr>
<td>Development</td>
<td>0979</td>
</tr>
<tr>
<td>Education</td>
<td>0250</td>
</tr>
<tr>
<td>Hospital Management</td>
<td>0679</td>
</tr>
<tr>
<td>Human Development</td>
<td>0738</td>
</tr>
<tr>
<td>Immunology</td>
<td>0972</td>
</tr>
<tr>
<td>Medicine and Surgery</td>
<td>0574</td>
</tr>
<tr>
<td>Mental Health</td>
<td>0579</td>
</tr>
<tr>
<td>Nutrition</td>
<td>0569</td>
</tr>
<tr>
<td>Obstetrics and Gynecology</td>
<td>0380</td>
</tr>
<tr>
<td>Occupational and Health Therapy</td>
<td>0354</td>
</tr>
<tr>
<td>Ophthalmology</td>
<td>0381</td>
</tr>
<tr>
<td>Pharmacology</td>
<td>0519</td>
</tr>
<tr>
<td>Pharmacy</td>
<td>0519</td>
</tr>
<tr>
<td>Physical Therapy</td>
<td>0392</td>
</tr>
<tr>
<td>Scientific Health and Nutrition</td>
<td>0739</td>
</tr>
<tr>
<td>Statistics</td>
<td>0463</td>
</tr>
<tr>
<td>Applied Sciences</td>
<td>0346</td>
</tr>
<tr>
<td>Applied Mechanics</td>
<td>0346</td>
</tr>
<tr>
<td>Computer Science</td>
<td>0974</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LIST OF SUBJECTS</th>
<th>0544</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ancient</td>
<td>0579</td>
</tr>
<tr>
<td>Medieval</td>
<td>0581</td>
</tr>
<tr>
<td>Modern</td>
<td>0582</td>
</tr>
<tr>
<td>Black</td>
<td>0378</td>
</tr>
<tr>
<td>African</td>
<td>0331</td>
</tr>
<tr>
<td>Asia, Australia and Oceania</td>
<td>0332</td>
</tr>
<tr>
<td>Canadian</td>
<td>0334</td>
</tr>
<tr>
<td>European</td>
<td>0334</td>
</tr>
<tr>
<td>Latin American</td>
<td>0336</td>
</tr>
<tr>
<td>Middle Eastern</td>
<td>0333</td>
</tr>
<tr>
<td>United States</td>
<td>0337</td>
</tr>
<tr>
<td>History of Science</td>
<td>0578</td>
</tr>
<tr>
<td>Law</td>
<td>0398</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>POLITICAL SCIENCE</th>
<th>0615</th>
</tr>
</thead>
<tbody>
<tr>
<td>General</td>
<td>0615</td>
</tr>
<tr>
<td>International Law and Relations</td>
<td>0616</td>
</tr>
<tr>
<td>Public Administration</td>
<td>0617</td>
</tr>
<tr>
<td>Recreation</td>
<td>0618</td>
</tr>
<tr>
<td>Social Welfare</td>
<td>0452</td>
</tr>
<tr>
<td>Sociology</td>
<td>0452</td>
</tr>
<tr>
<td>General</td>
<td>0626</td>
</tr>
<tr>
<td>Criminology and Penalogy</td>
<td>0627</td>
</tr>
<tr>
<td>Demography</td>
<td>0388</td>
</tr>
<tr>
<td>Ethnic and Racial Studies</td>
<td>0631</td>
</tr>
<tr>
<td>Individual and Family Studies</td>
<td>0628</td>
</tr>
<tr>
<td>Industrial and Labor</td>
<td>0628</td>
</tr>
<tr>
<td>Relations</td>
<td>0629</td>
</tr>
<tr>
<td>Public and Social Welfare</td>
<td>0630</td>
</tr>
<tr>
<td>Social Structure and Development</td>
<td>0700</td>
</tr>
<tr>
<td>Theory and Methods</td>
<td>0700</td>
</tr>
<tr>
<td>Transportation</td>
<td>0700</td>
</tr>
<tr>
<td>Urban and Regional Planning</td>
<td>0799</td>
</tr>
<tr>
<td>Women's Studies</td>
<td>0453</td>
</tr>
</tbody>
</table>
ABSTRACT

By Yaling Cai

This dissertation presents a study of fuzzy inference networks for pattern recognition problems. In this research, fuzzy neurons are defined and five types of fuzzy neurons are introduced. Three fuzzy inference models for pattern recognition systems, min-max inference model, min-sum inference model, and min-competitive inference model, are developed. Fuzzy inference networks based on the inference models and their learning algorithms are presented. The proposed fuzzy inference networks can learn fuzzy inference rules directly from training data. Two of the proposed fuzzy inference networks, Min-Max Fuzzy Inference Network and Min-Sum Fuzzy Inference Network, are applied to pattern classification problems. These two networks can learn the membership functions of all the classes and find out the soft and hard partitions according to the membership values. Another two fuzzy inference networks based on a min-competitive inference method are developed for invariant pattern recognition systems. These two Min-Competitive Fuzzy Inference Networks have been constructed for 2-D visual pattern recognition problems and have been tested with letter patterns with black and white pixel values. The learning speed of the proposed fuzzy inference networks is very fast. The structures of the proposed fuzzy inference networks are simple and they perform well when used in pattern classification and pattern recognition problems.
To my parents: Zhong-Qiong LI and Xu-yu CAI

and

to my husband Bin Zhang and my daughter Sylvia Zhang

for their love and support
ACKNOWLEDGEMENTS

I wish to express my grateful appreciation to Dr. H. K. Kwan, who is my advisor, for his inspiration, guidance, support, and encouragement throughout the study, and for financial support through the Natural Science and Engineering Research Council (NSERC) of Canada. I would also like to express my sincere gratitude to the external examiner, Dr. A. N. Venetsanopoulos of the Department of Electrical and Computer Engineering, University of Toronto, my department readers, Professor P. H. Alexander and Dr. J. J. Soltis, and my outside department reader, Dr. J. Morrisey of the School of Computer Science, for their valuable advice.

I would like to thank Dr. G. R. G. Raju, the head of the department, for his encouragement in my Ph.D. study. I would also like to acknowledge the University of Windsor for awarding me the University of Windsor Tuition Scholarship for two years.
# TABLE OF CONTENTS

**ABSTRACT** ......................................................... iii

**DEDICATION** ................................................... iv

**ACKNOWLEDGEMENTS** ........................................... v

## CHAPTER

### I  INTRODUCTION ............................................. 1

1.1 Review of Neural Networks .................................. 1
1.2 Review of Fuzzy Systems ................................... 8
1.3 Review of Fuzzy Neural Networks ......................... 16
1.4 Motivations and Objectives of The Research ............ 26

### II  FUZZY NEURONS ............................................ 29

2.1 Some Results from Neurobiology ......................... 30
2.2 Fuzzy Neurons ............................................. 34
2.3 Five Fuzzy Neuron Algorithms ............................ 39

### III  FUZZY INFERENCE NETWORKS (FINs) .................. 43

3.1 Fuzzy Inference Models for Pattern Recognition ........ 43
3.2 The Structures and Algorithms of Fuzzy Inference Networks ... 52
3.3 The Learning Strategy of Fuzzy Inference Networks ....... 61
3.4 Capabilities of Fuzzy Inference Networks ............... 68

### IV  FUZZY INFERENCE NETWORKS FOR PATTERN CLASSIFICATION 75
CHAPTER I

INTRODUCTION

1.1 Review of Neural Networks

Human beings always want to build intelligent machines. A good strategy to do so is to study how human beings think and act. One feature of the brain is the massive hardware parallelism: it uses many neurons -- the brain's basic computing elements -- to process information. This is different from conventional digital computers, which use only one computing engine, the Central Processing Unit (CPU). Only in the last decade has parallelism been introduced into digital computers, where several CPUs are working at the same time on the same problem. However, it is difficult to coordinate and program multiple CPUs of a traditional type of digital computers. The development of artificial intelligence based on the digital computer is not satisfactory because it is difficult, if not impossible, for digital computers to imitate the human being's abilities in perception, vision, associative memory, and pattern recognition [Anderson and Rosenfeld, 1988].
Neural networks are systems that are deliberately constructed to make use of some principles that are found in the human brain in the hope of achieving human-like performance. Neural networks have faster responses and higher performances than those of the sequential digital computers in emulating the capabilities of the human brain. A typical neural network has a massively parallel structure that is composed of many processing elements connected to each other through weights. Generally speaking, neural networks can be classified into feedforward structures and feedback structures. Feedforward neural networks are layered networks. There is an input layer, an output layer, and one or more hidden layers. The neurons in each layer only accept signals from neurons in the layer before. The processing of this type of networks is clear and easy to understand. Feedforward networks can be easily implemented in VLSI. However, many neurons and many training samples are needed for feedforward networks when dealing with complex problems. In feedback neural networks, each neuron is connected to all the other neurons. The signals are transferred between the neurons many times and the state of the network keeps changing. The network will achieve a stable state at the end. These types of networks can be viewed as nonlinear dynamic systems. The number of neurons needed in a feedback network are less than those in a feedforward network, but a feedback network needs a relatively long time to converge and the analysis of feedback
networks is complicated.

The model of an artificial neuron was first proposed by McCulloch and Pitts in 1943 [McCulloch and Pitts, 1943]. In 1949, Hebb suggested a learning rule for neurons [Hebb, 1949]. However, the neuron model did not interest scientists until the perceptron rule was published by Rosenblatt in 1958 [Rosenblatt, 1958]. In the years following this discovery until the end of the 1960s, many techniques were developed to investigate the properties of the perceptron and its applications [Widrow and Hoff, 1960, Block, 1962, Minsky and Papert, 1969]. After that, the perceptrons were still unable to attract the scientists because of the fast development of digital computers. This situation for neural network research had not changed until the early 1980s, then Hopfield and others introduced outer product rules as well as equivalent approaches based on the early work of Hebb for training a class of feedback networks and pointed out that their neural networks could be implemented by VLSI [Hopfield, 1982]. Since then, a number of developments have occurred. Grossberg and Carpenter developed Adaptive Resonance Theory (ART) [Grossberg 1976, Grossberg 1980, Carpenter and Grossberg, 1988]. Fukushima explored Neocognitron models [Fukushima et al., 1983]. Kohonen proposed a self-organizing system on feature maps [Kohonen, 1982]. Some other significant models of feedback networks
include Kosko's Bidirectional Associative Memory [Kosko, 1987] and Boltzmann Machine proposed by Ackley et al. [Ackley et al., 1985]. While the development of feedback networks was taking place, feedforward structures travelled an independent path. Widrow et al. developed the Madaline I and II and Adline [Widrow and Hoff, 1960; Widrow 1987; Widrow and Winter, 1988]. Rumelhart et al. proposed a backpropagation network, which is widely used nowadays [Rumelhart et al., 1986]. The research work on neural networks before the middle of the 1980s concentrated on the neuron models, network structures, and learning algorithms, which formed the basic theories of neural networks. After that, many efforts were devoted to the implementation of neural networks and the application of neural networks in various fields. Many software and hardware neural network products have come out.

The discipline of neural networks has grown rapidly in recent years. Many models have been presented and the applications of neural networks have been successful. In the fields of continuous speech recognition, image pattern recognition, sonar and radar signal processing, uncertain knowledge processing, and adaptive control systems, neural networks have been proved to be useful.

Neural networks store patterns with distributed coding and they are trainable nonlinear dynamic systems. Neural networks can perform pattern
matching tasks, while traditional computer architectures, however, are inefficient at these tasks. Neural networks are model-free estimators. They do not make assumptions of how outputs depend on inputs. Instead, they adjust themselves to a given training set by a learning algorithm. Neural networks are excellent at emulating human-like systems that can process the same type of information as our brain processes, such as image and speech processing, robotics, non-model based control, and decision making systems.

The major advantages of neural networks are:

(1) The structures of neural networks are adaptive to the environments and so they possess the ability to learn from environments;

(2) Neural networks are generally represented by nonlinear differential equations. They are nonlinear dynamic systems and may have complicated dynamic properties;

(3) Neural networks can solve those problems that have very complicated environments, unclear reasoning rules, variable or conflicting information sources;

(4) Neural networks have powerful computing abilities and very high speed via massive parallelism structures;
Neural networks have fault tolerance because they are constructed from a large amount of neurons. They are not sensitive to missing, confusing or noisy data.

Neural networks are simplified models of the human brain. The properties of neurons and the connecting topology between neurons are simplified to imitate the computational ability of the human brain. Consequently, neural networks have some features of the human brain. The simplification of neural network models makes it possible for them to be implemented. However, many important results of biological neural science are not included in neural network models. The loss of modelling of some important features of biological neural systems will result in the loss of some important functions of the human brain. The modelling of the human brain should make use of the results of biological neural science as much as possible while considering the possibility of implementation based on present technology. Current computer technology potentially allows us to build a system of a complexity that approaches the number of elements and interconnections of the human brain. So, it is necessary and possible to improve neuron models of neural networks to build more intelligent neural network systems.
On the other hand, the present neural networks represent knowledge in an implicit form. The knowledge is represented in the connections among the neurons of neural networks. The interpretation of the meaning of a network is the most difficult and critical task in neural networks because of the lack of an appropriate language to transcribe the knowledge into an explicit form. In this research, a new type of neural network is developed to overcome this problem. Fuzzy set theory is introduced into the neural network and the knowledge represented in the connections of the network is described by fuzzy inference rules. In the next section, the basic ideas and features of fuzzy systems will be discussed.
1.2 Review of Fuzzy Systems

Fuzzy sets were introduced in 1965 by L. A. Zadeh as a new way to represent fuzziness in everyday life [Zadeh, 1965]. What is called fuzziness is one aspect of uncertainty. Fuzziness is the ambiguity that can be found in the definition of a concept or the meaning of a word. For example, the uncertainty in expressions like "old person", "high temperature", or "small number" can be called fuzziness.

In traditional mathematics, probability theory has been the only one that has dealt with uncertainty. The uncertainty generally relates to the occurrence of phenomena, as symbolized by the concept of randomness. Randomness and fuzziness are different aspects of uncertainty. The uncertainty in a random event can be clarified by the passage of time or testing. However, the ambiguity lies in the meaning of the words. It always follows them since it is an essential characteristic of the words [Terano et al., 1992].

Probability theory has a long history and is widely used in engineering and natural sciences. Fuzzy theory was developed 30 years ago. Its use is not yet
widespread, but fuzziness expresses much more everyday uncertainty than probability. This is because fuzziness expresses the uncertainty that can be found in the meanings of many words of our everyday life and words are indivisible from humans' thinking. All people think and transmit their thoughts and information by means of words. So fuzziness is a kind of uncertainty that is involved very much with our everyday life.

It is believed that the effectiveness of the human brain comes not only from precise cognition, but also from fuzzy concepts, fuzzy judgment and fuzzy reasoning. The fuzzy processing feature of the human brain includes two aspects. On the one hand, the input information contains some fuzziness. On the other hand, the human brain fuzzifies the accepted information and processes it using a fuzzy method. The outstanding feature of fuzzy theory is the ability to express the fuzziness in a comparatively undistorted manner. This feature makes it possible to establish fuzzy systems that can process fuzzy information using fuzzy algorithms and behave more similarly to human thinking than traditional computers.

Traditional set theory describes crisp events, events that either do or do not occur. There is no middle ground. In contrast, fuzzy set theory measures
the degree to which an event occurs. The definition of a fuzzy set is:

A fuzzy set $A$ is a subset of the universe of discourse $X$ that admits partial membership. The fuzzy set $A$ is defined as the ordered pair $A = \{x, m_A(x)\}$, where $x \in X$, and $0 \leq m_A(x) \leq 1$. The membership function $m_A(x)$ describes the degree to which the object $x$ belongs to the set $A$, where $m_A(x) = 0$ represents no membership, and $m_A(x) = 1$ represents full membership.

As an example, let $X$ represent the ages of all people. The subset $A$ of $X$ that represents those people who are young is a fuzzy set with the membership function shown in Fig. 1.1 [Simpson, 1993].

![Fig. 1.1 The membership function of the fuzzy set A that represents those people who are young among all the people](image_url)
Let us look at another fuzzy set $F$ of real numbers that are close to 5. Since the property "close to 5" is fuzzy, there is not a unique membership function for $F$. Rather, the modeller must decide, based on the potential application and properties desired for $F$, what $m_F$ should be. One of the biggest differences between crisp and fuzzy sets is that the former always have unique membership functions (whose value is either 1 or 0), whereas every fuzzy set has an infinite number of membership functions that may represent it. Fig. 1.2 gives two membership functions that represent the fuzzy number "close to 5" [Bezdek and Pal, 1992].

![Membership functions of the fuzzy set $F$](image)

**Fig. 1.2** Membership functions of the fuzzy set $F$ that represent "real numbers close to 5"
The operations on fuzzy sets are extensions of those used for traditional sets. The common operations of fuzzy sets include union, intersection, complement, comparison, and containment, [Pal and Majumder, 1986]. Assuming $X$ is the universe of discourse, $A \in X$, and $B \in X$, these operations are defined as follows:

**Union** $(A \cup B)$:

$$m_{A \cup B}(x) = \max(m_A(x), m_B(x)), \ \forall x \in X$$

**Intersection** $(A \cap B)$:

$$m_{A \cap B}(x) = \min(m_A(x), m_B(x)), \ \forall x \in X$$

**Complement** $(\bar{A})$:

$$m_{\bar{A}}(x) = 1 - m_A(x), \ \forall x \in X$$

**Equivalence Relation**:

$A = B$ iff $m_A(x) = m_B(x), \ \forall x \in X$

**Containment**:

$A \subseteq B$ iff $m_A(x) < m_B(x), \ \forall x \in X$

In addition to these operations, the following properties hold for fuzzy sets:
Commutativity: \[ A \cup B = B \cup A, \quad A \cap B = B \cap A \]

Distributivity: \[ A \cup (B \cup C) = (A \cup B) \cup C, \]
\[ A \cap (B \cap C) = (A \cap B) \cap C \]

De Morgan's Laws:
\[ \overline{A \cup B} = A \cap \overline{B}, \quad \overline{A \cap B} = \overline{A} \cup \overline{B} \]

Bounded Sum \(A \oplus B\):
\[ m_{A \oplus B}(x) = \min(1, m_A(x) + m_B(x)), \quad \forall x \in X \]

Bounded Difference \(A \ominus B\):
\[ m_{A \ominus B}(x) = \max(0, m_A(x) - m_B(x)), \quad \forall x \in X \]

Algebraic Product \(A \odot B\):
\[ m_{A \odot B}(x) = m_A(x) \cdot m_B(x), \quad \forall x \in X \]

Algebraic Sum \(A + B\):
\[ m_{A + B}(x) = m_A(x) + m_B(x) - m_A(x) \cdot m_B(x), \quad \forall x \in X \]

It should be mentioned here that there could be many definitions for the basic operations of fuzzy sets. For examples, union may be defined as:
\[ m_{A \cup B}(x) = \min(1, m_A(x) + m_B(x)), \quad \forall x \in X, \]
and intersection may be defined as:
\[ m_{A \cap B}(x) = \max(0, m_A(x) + m_B(x) - 1), \quad \forall x \in X. \]
Fuzzy systems use fuzzy set theory to deal with fuzzy or nonfuzzy information. Among all the fuzzy systems, fuzzy rule-based systems are the most important ones and are successfully used in many problems such as decision making systems, expert systems and control systems. Fig. 1.3 shows a typical fuzzy-rule based system. A fuzzy rule base and inference engine are the core of the fuzzy-rule based system. A fuzzifier and defuzzifier are used when needed. A fuzzy rule base can be expressed by a set of fuzzy inference rules in the form of "IF x is A THEN y is B". The inference engine implements a fuzzy inference algorithm that deduces results from the inference rules and present inputs. Fuzzy inference methods are based on fuzzy logic in consideration of the special features for practical systems.

![Diagram of a fuzzy-rule based system]

Fig. 1.3 A typical fuzzy-rule based system
Fuzzy systems store "rules" and estimate sampled functions from linguistic inputs to generate linguistic outputs. Fuzzy methods are powerful for modelling human thinking and perception. The application of fuzzy theory is firmly tied to human thinking and behaviour. It is successfully used to express human experiences in a form that a machine can use, to imitate human pattern recognition or judgment capability, to make models of human feelings or languages, to convert information into a form that people can easily understand, to compress large amounts of information, to make models of human behaviour, and to make models of social systems. However, at present, the design of membership functions and fuzzy inference rules is the bottleneck of the procedure for the design of fuzzy rule-based systems. The most straightforward approach is to define membership functions and rules subjectively by studying a human-operated system or an existing dynamic system and then testing the design for the proper output. The membership functions and the rules should be adjusted if the design fails the test. This procedure needs a lot of manual work. If a fuzzy system has learning ability, the membership functions and the rules can be determined automatically through a training procedure. Fuzzy neural networks are such systems that integrate the learning ability of neural networks into fuzzy systems. In the next section, fuzzy neural systems will be discussed.
1.3 Review of Fuzzy Neural Systems

The theories of neural networks and fuzzy sets provide two complementary ways of modelling the human brain. At one level, the neural network provides a means for modelling the lower level processes of the human brain, that is, the physiology of the brain. At the other extreme, fuzzy theory provides a means of capturing the higher level of human thought processes as well as emergent properties that can be seen as a psychologic modelling of the mind [Kosko, 1992]. In this sense, neural networks are related to the "hardware" of the brain, while fuzzy logic is involved with the "software" of the brain; however, they are strongly associated to provide a description of the basic behaviour of the human mind. Since these two approaches generally attack the design of "intelligent" systems from quite different angles, the marriage of neural networks and fuzzy logic has a sound technical basis. Consequently, the combination of neural networks and fuzzy logic, which is called fuzzy neural networks, could result in a useful tool for exploring the function of the human brain. Fuzzy neural network systems might turn out to be "user friendly" since they work like us. Fuzzy set theory has proved itself to be of significance in pattern recognition problems and neural networks are excellent at pattern recognition.
So fuzzy neural networks could find an important role in this field.

The potential advantages of fuzzy neural networks include:

(1) Integrating the advantages of both fuzzy logic and neural networks;
(2) Interpreting the meaning of the network using fuzzy theory;
(3) Incorporating learning ability into fuzzy systems and providing a systematic method of designing fuzzy systems;
(4) Efficient hardware usage and high speed because of distributed representation of knowledge using fuzzy theory.

In the last several years, there have been some research efforts aimed at synthesizing fuzzy logic with neural networks. The research on fuzzy neural systems has two directions: one is to employ the learning ability of neural networks as a tool to design fuzzy logic systems, the other is to develop fuzzy neural networks to deal with information in neural network structures while using fuzzy methods. Some important work that has been done in this field is listed in TABLE 1.1.
<table>
<thead>
<tr>
<th>Year</th>
<th>Researchers</th>
<th>Systems</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>1975</td>
<td>S. C. Lee</td>
<td>Fuzzy neural network</td>
<td>Fuzzy language recognition</td>
</tr>
<tr>
<td></td>
<td>E. T. Lee</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>J. Keller</td>
<td>Fuzzy perceptron</td>
<td>Classification</td>
</tr>
<tr>
<td></td>
<td>D. Hunt</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1985</td>
<td>T. Yamakawa</td>
<td>Fuzzy neuron model for pattern recognition</td>
<td>Handwritten recognition</td>
</tr>
<tr>
<td></td>
<td>S. Tomoda</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1986</td>
<td>B. Kosko</td>
<td>Fuzzy associative memory</td>
<td>Fuzzy control system</td>
</tr>
<tr>
<td>1989</td>
<td>R.J. Machado</td>
<td>Combinatorial neural model</td>
<td>Expert systems</td>
</tr>
<tr>
<td></td>
<td>A.F. Rocha</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1990</td>
<td>H. Takagi</td>
<td>NN-driven fuzzy reasoning system</td>
<td>Fuzzy reasoning</td>
</tr>
<tr>
<td></td>
<td>I. Hayashi</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1991</td>
<td>N. P. Archer</td>
<td>Monotonic function neural network model</td>
<td>Classification</td>
</tr>
<tr>
<td></td>
<td>S. Wang</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1991</td>
<td>C. T. Lin</td>
<td>Connectionist model for fuzzy logic control and decision system</td>
<td>Fuzzy control and decision systems</td>
</tr>
<tr>
<td></td>
<td>C. S. G. Lee</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1992</td>
<td>A. F. Rocha</td>
<td>Modular neural networks</td>
<td>Fuzzy control and expert systems</td>
</tr>
<tr>
<td></td>
<td>R. Yager</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1992</td>
<td>L. X. Wang</td>
<td>Backpropagation fuzzy system</td>
<td>Identification of nonlinear systems</td>
</tr>
<tr>
<td></td>
<td>J. M. Mendel</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Year</td>
<td>Author(s)</td>
<td>Method</td>
<td>Field</td>
</tr>
<tr>
<td>------</td>
<td>--------------------</td>
<td>-------------------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>1992</td>
<td>G. A. Carpenter</td>
<td>Fuzzy ARTMAP</td>
<td>Classification</td>
</tr>
<tr>
<td></td>
<td>S. Grossberg</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>N. Markuzon</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>J. H. Reynolds</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>D. B. Rosen</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1992</td>
<td>P. K. Simpson</td>
<td>Fuzzy min-max neural</td>
<td>Classification and Clustering</td>
</tr>
<tr>
<td></td>
<td></td>
<td>network</td>
<td></td>
</tr>
<tr>
<td>1993</td>
<td>W. Pedrycz</td>
<td>Fuzzy-set oriented</td>
<td>Knowledge processing</td>
</tr>
<tr>
<td></td>
<td>A. F. Rocha</td>
<td>neurons</td>
<td></td>
</tr>
<tr>
<td>1993</td>
<td>Y. H. Kuo</td>
<td>Connectionist fuzzy</td>
<td>Classification</td>
</tr>
<tr>
<td></td>
<td>C. I. Kao</td>
<td>classifier</td>
<td></td>
</tr>
<tr>
<td></td>
<td>J. J. Chen</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>fuzzy inference system</td>
<td></td>
</tr>
<tr>
<td>1994</td>
<td>C. M. Higgins</td>
<td>Fuzzy rule-based networks</td>
<td>Fuzzy control systems</td>
</tr>
<tr>
<td></td>
<td>R. M. Goodman</td>
<td>for control</td>
<td></td>
</tr>
<tr>
<td>1994</td>
<td>W. Pedrycz</td>
<td>Generalized fuzzy Petri net</td>
<td>System modelling and analysis</td>
</tr>
<tr>
<td></td>
<td>F. Fernando</td>
<td>model</td>
<td></td>
</tr>
<tr>
<td>1995</td>
<td>S. Mitra</td>
<td>Fuzzy multi-layer</td>
<td>Fuzzy expert system</td>
</tr>
<tr>
<td></td>
<td>S. K. Pal</td>
<td>perceptron</td>
<td></td>
</tr>
</tbody>
</table>

The first idea of fuzzy neural networks was proposed by Lee et al. in 1975 [Lee et al., 1975]. They extended the McCulloch-Pitts neuron model to a more general model which allows the activity of the neuron to be a fuzzy process. They investigated the application of their fuzzy neural network to the synthesis of fuzzy automata and showed that any n-state minimal fuzzy automaton can be realized by a network of m fuzzy neurons, where \( \log_2 n \leq m \leq 2n \). An example of
the realization of a fuzzy language recognizer using a fuzzy neural network was also presented. However, they did not study the learning capability of their fuzzy neural networks.

Keller and Hunt introduced fuzzy membership functions into the perceptron algorithm in 1985 [Keller and Hunt, 1985]. Their fuzzy perceptron converges quickly in the separable case and improves the convergence in the non-separable case. A method of generating membership functions is developed for their fuzzy perceptron.

In 1986, Yamakawa and Tomoda described a simple fuzzy neuron model and used it in a neural network for the application of handwritten character recognition problems [Yamakawa and Tomoda, 1986]. However, they did not describe the specific learning algorithm for this network.

Kosko presented the fuzzy associative memory model and suggested Hebb's rule to train it in 1989 [Kosko, 1989]. The fuzzy associative memories use fuzzy matrices instead of neurons to represent mapping between fuzzy sets. Fuzzy associative memories have been applied in fuzzy control systems and fuzzy expert systems.
Machado and Rocha have constructed a fuzzy connectionist expert system using a combinatorial neural model (CNM) in 1990 [Machado and Rocha, 1990]. The CNM uses fuzzy numbers, fuzzy-AND neurons and fuzzy-OR neurons to classify patterns. Hebb’s rule is used to train the network.

In 1990, Takagi and Hayashi proposed a neural network driven fuzzy reasoning system. In that system, the membership functions of fuzzy inference rules can be formed by employing feedforward neural networks [Takagi and Hayashi, 1990]. Takagi et al. have also used this structure in pattern recognition problems and it has better performance than ordinary neural networks [Takagi et al., 1990]. However, it is complicated to train this system as it contains several feedforward neural networks.

Archer and Wang developed a monotonic function neural network model and used it to represent fuzzy membership functions in two-class pattern recognition problems in 1991 [Archer and Wang, 1991]. This model used several small neural networks to construct the boundary of classes. Its applications are limited because it can only deal with two-class monotonic classifications.

Lin and Lee proposed a connectionist model for fuzzy logic control and
decision system in 1991 [Lin and Lee, 1991]. This fuzzy control/decision network can be constructed from training samples by machine learning techniques and the neural network structure can be trained to develop fuzzy logic rules and find optimal bell-shaped input/output membership functions. A backpropagation type learning algorithm was used in this approach.

Rocha and Yager investigated the close connection between fuzzy set theory and neural networks in 1992 [Rocha and Yager, 1992]. A new structure for the neuron was proposed and modular neural networks (MNN) were constructed. MNN can be used to implement adaptive fuzzy controllers and expert systems.

Jang developed an adaptive-network-based fuzzy inference system (ANFIS) which is used as a fuzzy controller in 1992 [Jang, 1992]. This ANFIS employs the backpropagation learning procedure to adjust the parameters of the bell-shaped fuzzy membership functions of the input variables.

Wang and Mendel proposed a backpropagation fuzzy system and used it as a nonlinear dynamic system identifier in 1992 [Wang and Mendel, 1992]. They showed that fuzzy systems can be viewed as a three-layer feedforward
network and developed a backpropagation algorithm to train them to match desired input-output pairs by adjusting the parameters of Gaussian membership functions.

Carpenter et al. combined fuzzy set theory with the ART algorithm and proposed a fuzzy ART structure for multidimensional maps [Carpenter et al., 1992]. The Fuzzy ARTMAP realized a min-max learning rule that minimizes predictive error and maximizes code compression. Simulations of this fuzzy ARTMAP on classifications were performed and compared with other classification algorithms. However, preprocessing is needed to normalize the input vectors in the fuzzy ARTMAP.

Simpson proposed fuzzy min-max neural networks and their learning algorithms for classification and clustering [Simpson, 1992 and 1993]. The fuzzy min-max classifier used fuzzy sets as pattern classes and each fuzzy set is an aggregation of fuzzy set hyperboxes. The fuzzy min-max clustering neural network stabilizes into pattern clusters in only a few passes through a data set. Fuzzy as well as hard cluster boundaries can be obtained by the network. The fuzzy min-max networks provide a degree of membership information that is extremely useful in higher level decision making and information processing.
Pedrycz and Rocha introduced and studied different fuzzy-set oriented computational models of neurons [Pedrycz and Rocha, 1993]. The generic topologies of the neurons emerging there are significantly influenced by basic operators (AND, OR, NOT) encountered in the theory of fuzzy sets. The two broad categories of neurons embrace aggregation neurons (named AND and OR neurons) and referential processing units. The proposed neurons were used to construct knowledge-based networks and the learning of the networks was also discussed.

Kuo et al. proposed a connectionist fuzzy classifier based on a four layer feedforward neural network and discussed its hardware implementation [Kuo et al., 1993]. This connectionist fuzzy classifier realizes the weighted "Euclidean distance" fuzzy classification concept in a massively parallel manner to recognize input patterns. It employs a hybrid supervised/unsupervised learning algorithm.

Sun proposed an adaptive-network-based fuzzy inference system in 1994 [Sun, 1994]. This approach summarized Jang’s architecture of employing an adaptive network and the Kalman filtering algorithm to identify the system parameters. A data structure, called fuzzy binary boxtree, was introduced to organize rules so that the rule base matched against input signals with logarithmic
efficiency. This scheme can be used in various situations of pattern representation or data expression.

Higgins and Goodman proposed fuzzy rule-based networks for learning fuzzy membership functions and rules for control problems [Higgins and Goodman, 1994]. The approach contains three steps: learning membership functions and creating rules; simplifying the rules; and constructing a computational neural network to represent the obtained fuzzy control system.

Pedrycz and Fernando presented a generalized fuzzy Petri net model which has particular interest for handling fuzziness in system modelling and analysis [Pedrycz and Fernando, 1994]. Fuzzy set operators are added to the two-value Petri net and the new fuzzy Petri net model has learning abilities.

Mitra and Pal developed a fuzzy multi-layer perceptron fuzzy expert system [Mitra and Pal, 1995]. It infers the outputs class membership values of an input pattern and generates a measure of certainty expressing confidence in the decision. The system is used in the speech recognition problem.
1.4 Motivations and Objectives of the Research

The attempt at building fuzzy neural systems has achieved some successes. However, the existing systems are complex and their learning algorithms are not efficient. Most of the existing fuzzy neural systems use backpropagation type learning algorithms to adjust their parameters. The main problem with backpropagation learning is that it needs many training epochs to converge and it takes a long time to train a network for a large set of training data.

Also, there is not yet a fuzzy neural structure which can effectively deal with visual pattern recognition problems. Current fuzzy neural systems that can be used as classifiers cannot be applied directly to visual pattern recognition problems. They require a preprocessing of the input patterns to extract important features from the original data. However, it is difficult or impossible to do preprocessing in some real-time applications. So it is inconvenient to develop efficient fuzzy neural network structures that can deal with visual pattern recognition problems without preprocessing of the input data and that has the advantages of both fuzzy systems and neural networks.
The motivations of this research are: to establish intelligent fuzzy rule-based neural network systems for pattern recognition problems, and to develop efficient learning algorithms for the proposed fuzzy neural network systems.

The objectives of this research are: to combine the advantages of both neural networks and fuzzy systems, to develop fuzzy inference methods and networks for pattern recognition systems, to investigate the features of the proposed fuzzy inference networks, and to develop fuzzy inference networks that can deal with visual pattern recognition problems without preprocessing of the input data.

In Chapter 2, fuzzy neurons are defined and five types of fuzzy neurons are introduced. In Chapter 3, fuzzy inference methods for pattern recognition problems: Min-Max inference method, Min-Sum inference method, and Min-Competitive inference method are developed and three fuzzy inference networks based on these models are constructed using fuzzy neurons. In Chapter 4, two fuzzy inference networks based on Min-Max and Min-Sum inference methods are applied to pattern classification problems. In Chapter 5, fuzzy inference networks that are based on the Min-Competitive inference method and which do not need preprocessing of the input data are proposed for visual pattern recognition
problems. Chapter 6 gives the conclusions of this research and suggestions to future work.
CHAPTER II

FUZZY NEURONS

Neurons are the information processing units in the human brain. In most of the present artificial neural networks, neurons are simplified as threshold units; each of them has only two states, 0 or 1. In the human brain, there are at least ten billion nerve cells which belong to more than two hundred types. Each type of nerve cell has different features and different functions. Moreover, the interactions between nerve cells are very complicated electrochemical processes. The two-state threshold units with linear weights are too simple to express the features of neurons and their interactions.

In this chapter, the properties of neurons in the human brain will be investigated and the model of fuzzy neurons will be proposed. The definition of fuzzy neurons is an extension of the previous artificial neurons. Fuzzy neurons are more complicated but they can be implemented by the present technology.

29
2.1 Some Results from Neurobiology

There are two kinds of cells in the human brain. One is called a nerve cell and the other is called a glair cell. The nerve cells process information while glair cells support nerve cells and make contact with the outside. There are huge amounts and many types of nerve cells in the human brain. Each type of nerve cell has a different structure and function.

A typical nerve cell (as shown in Fig. 2.1) is composed of four parts, the cell body, the dendrites, the axon, and the synapses (terminal fibres of the axon) [Anderson and Rosenfeld, 1988]. The nerve cell receives inputs from other neurons through dendrites. Then the cell body integrates the inputs and generates nerve impulses as an output, which is then sent to other neurons by the axon and synapses. A single neuron can receive on the order of hundreds or thousands of input lines and may send its output to a similar number of other neurons. The axon of a nerve cell can be long or short, and there may not even be an axon for a nerve cell. Many axons are insulated by a myelin sheath interrupted at intervals by the regions known as nodes of Ranvier. Synapses come in a number of different forms,
Fig. 2.1 A typical nerve cell of a vertebrate animal
which determine how the nerve cell affects other nerve cells. The two basic types of synapses are: excitatory synapses, which make it more likely that the neuron receiving them will fire action potentials, and inhibitory synapses, which make it less likely that the neuron receiving them will fire action potentials.

A neuron is a complex electrochemical device that contains a continuous internal potential called a membrane potential. When the signals arrive at the dendrites of a nerve cell, some ions with electric charges will be generated and these ions will lead to a change in its membrane potential. When the membrane potential exceeds a threshold, the nerve cell can propagate impulses down its axon and synapses to other nerve cells. The outputs of biological neurons are continuous valued and the neuron acts like a voltage to frequency converter, converting membrane potential into firing rate.

In the information processing procedures of the human brain, the signals are not always transmitted by the axon. Sometimes, the cell body, or even the dendrites, can be the transmitter(s). Similarly, the signal receivers are not always the dendrites: the cell body, the synapses, or even the Ranvier nodes can accept information from other neurons. Because of the distributed nature of information processing of the neurons, it is unnecessary for some neurons to exceed the
thresholds in order to affect other neurons. Sometimes, even the thresholds vary with environments.

In order to imitate the function of the human brain, the models of neurons have to adapt the important results of biological neural science. The ignorance of some important details of biological neural systems will lead to the omission of some important functions of the human brain. At the same time, the modelling of the human brain should consider the possibility of implementation based on present technology. In the next sections, improved neuron models will be defined. The new neuron model will have a continuous state to represent the membrane, multiple outputs with nonlinear output functions will describe the complicated ways of the neuron affecting other neurons. Different types of neuron models also will be defined by choosing different aggregation functions including fuzzy logic operations.
2.2. Fuzzy Neurons

Neurons are the basic computing units of neural networks. In most of the previous researches, it is assumed that all the neurons in a network are identical. Typically, a neuron has N weighted inputs $x_i$ to $x_N$ and one output $y$. The neuron sums these weighted inputs and transfers the result to a nonlinear threshold function. That is:

$$z = \sum_{i=1}^{N} w_i \cdot x_i$$  \hspace{1cm} (2.1)

$$y = s = f(z - t)$$  \hspace{1cm} (2.2)

where $w_i$ ($i=1, 2, ..., N$) is the $i$th input weight, $z$ is the net input, $s$ is the state, $f(\cdot)$ is the activation function, and $t$ is the activating threshold.

In this research, the definition of a neuron is generalized to a more general model. Now a neuron has $N$ inputs $x_1$ to $x_N$ and $M$ outputs $y_1$ to $y_M$. Moreover, we have:
\[ z = h(w_1, x_1, w_2, x_2, \ldots, w_N, x_N) \]  \hspace{1cm} (2.3)

\[ s = f(z, t) \]  \hspace{1cm} (2.4)

\[ y_j = g_j(s) \quad j = 1, 2, \ldots, M \]  \hspace{1cm} (2.5)

where \( w_i \) (\( i = 1, 2, \ldots, N \)) is the \( i \)th input weight, \( z \) is the net input, \( s \) is the state, \( h() \) is the aggregation function, \( f() \) is the activation function, \( t \) is the activation threshold, and \{\( g_j() \mid j = 1, 2, \ldots, M \}\} represent the \( M \) output functions of the neuron.

The inputs are signals coming from other neurons and the outputs are signals sent to other neurons. The state \( s \) corresponds to the membrane potential of nerve cells. The aggregation function \( h() \) expresses the way that the neuron integrates the effects from all other neurons. The aggregation function for a traditional artificial neuron is a simple summation. The activation function \( f() \) describes how the neuron adjusts its membrane. The output functions give a more comprehensive description of the interactions between neurons because a
neuron affects different neurons in different ways. For example, some neurons are sensitive to the low membrane potential of a neuron but some are sensitive to the high membrane potential of the neuron. The output functions can be used to express the nonlinear interactions between neurons.

The input weights, output functions, and the activation threshold can be adjusted during the learning procedure. The aggregation function \( h(\cdot) \) and activation function \( f(\cdot) \) are the intrinsic features of a neuron. Many types of neurons can be defined by changing the functions \( h(\cdot) \) and \( f(\cdot) \).

The inputs of a neuron can be fuzzy variables or nonfuzzy variables. The aggregation operation \( h(\cdot) \) can be a fuzzy or nonfuzzy operation. The outputs of a neuron can be used to represent fuzzy concepts when the output functions \( \{g_j(\cdot) \mid j=1, 2, \ldots, M\} \) are membership functions of fuzzy sets.

A neuron is called a fuzzy neuron (FN) if it has the ability to cope with fuzzy information, i.e., if the inputs or the outputs are fuzzy variables, and the aggregation function \( h(\cdot) \) is a fuzzy operation such as \( \min, \max \), etc., or the output functions \( \{g_j(\cdot) \mid j=1, 2, \ldots, M\} \) are membership functions. Fig. 2.2 shows a fuzzy neuron.
From the previous discussion, we know that the input weights, the output functions and the activation threshold that describe the interactions between neurons, can be adjusted during the learning procedure. These parameters are called learning parameters. The process of adjusting the learning parameters is in fact a process of adjusting the structure of the fuzzy neural network.

Fig. 2.2 A fuzzy neuron
The differences between a fuzzy neuron and a traditional non-fuzzy neuron are:

(a) The aggregation function of a fuzzy neuron is a fuzzy operation, while that of a traditional non-fuzzy neuron is a simple summation.

(b) A fuzzy neuron has multiple outputs with different output functions while a traditional non-fuzzy neuron has one output that is equal to the state of the neuron.

(c) The learning of a fuzzy neuron can be performed by adjusting its weights, or activation threshold or output functions while a traditional non-fuzzy neuron can only adjust its weights and threshold.

(d) The inputs or the outputs of a fuzzy neuron are fuzzy variables and the fuzzy neuron can express and process fuzzy information while a non-fuzzy neuron can only deal with non-fuzzy information.

Fuzzy neurons can process fuzzy information and implement fuzzy systems in neural network structures. In the following section, five types of fuzzy neurons that will be used in later chapters are defined.
2.3 Five Fuzzy Neuron Algorithms

1. Transit Fuzzy Neuron (TRAN-FN): If a neuron has only one input $x$ such that

$$ z = s = x \quad (2.6) $$

$$ y_j = g_j(s) \quad j = 1, 2, \cdots, M \quad (2.7) $$

where the output functions $\{g_j(\ )\mid j = 1, 2, \ldots, M\}$ are membership functions, then it is called a Transit Fuzzy Neuron or a TRAN-FN. The outputs of a TRAN-FN are fuzzy variables. TRAN-FNs can be used to do fuzzification in a network.

2. Maximum Fuzzy Neuron (MAX-FN): If the inputs of a neuron are fuzzy variables and a maximum function (fuzzy OR operation) is used as the aggregation function of the neuron such that
\[ z = \max_{i=1}^{N} (w_i \cdot x_i) \]  

(2.8)

then it is called a Maximum Fuzzy Neuron or MAX-FN. The inputs and the outputs of a MAX-FN are fuzzy variables.

3. Minimum Fuzzy Neuron (MIN-FN): If the inputs of a neuron are fuzzy variables and a minimum function (fuzzy AND operation) is used as the aggregation function of the neuron such that

\[ z = \min_{i=1}^{N} (w_i \cdot x_i) \]  

(2.9)

then it is called a Minimum Fuzzy Neuron or MIN-FN. The inputs and the outputs of a MIN-FN are fuzzy variables.

4. Sum Fuzzy Neuron (SUM-FN): If the inputs of a neuron are fuzzy variables and a summation is used as the aggregation function of the neuron such that

\[ z = \sum_{i=1}^{N} (w_i \cdot x_i) \]
\[ z = \sum_{i=1}^{N} (w_i \cdot x_i) \]  

(2.10)

then it is called a Sum Fuzzy Neuron or SUM-FN. The inputs and the outputs of a SUM-FN are fuzzy variables.

5. Competitive Fuzzy Neuron (COMP-FN): If the inputs of a neuron are fuzzy variables and it has a variable threshold \( t \) and only one output such that

\[ z_i = h(w_i, x_i) \]  

(2.11)

\[ y = s = f(w_i, x_i), \quad \text{if} \quad z_i = t \]  

(2.12)

\[ t = r(c_1, c_2, \cdots, c_M) \]  

(2.13)

where \( z_i \) is the \( i \)th net input, \( r(\quad) \) is the threshold function, \( c_j (j = 1 \text{ to } M) \) are the threshold inputs, and the output \( y \) is determined by the net inputs \( z_i (i = 1, 2, \ldots, N) \) and the threshold, then it is called a Competitive Fuzzy Neuron or COMP-FN.
Many types of fuzzy neurons could be defined by choosing different kinds of aggregation function $h(\ )$ and/or activation function $f(\ )$. Fuzzy neurons are used to construct fuzzy neural networks. How to choose fuzzy neurons when constructing a fuzzy neural network is based on the features of the fuzzy neurons and the use of the fuzzy neural network. For example, a TRAN-FN can be used in the input layer of a fuzzy neural network to transfer non-fuzzy information (such as binary data) into fuzzy information (such as membership values). In practice, many types of fuzzy neural networks can be constructed for different applications. In the following chapters, fuzzy inference networks and their associated learning algorithms will be proposed.
CHAPTER III

FUZZY INFERENCE NETWORKS

In this chapter, fuzzy inference models for pattern recognition systems are developed. By using the fuzzy neurons introduced in Chapter 2, fuzzy inference networks are constructed based on the proposed inference methods. The learning ability and the features of the fuzzy inference networks are also discussed in this chapter.

3.1 Fuzzy Inference Models for Pattern Recognition

Fuzzy rule-based systems are successfully used in expert systems and control systems. However, there is not yet a fuzzy rule-based system for pattern recognition problems. In this section, fuzzy inference models for pattern recognition systems will be developed based on the existing fuzzy inference models by considering the special features of pattern recognition systems.
Let us first take a look at an example of a fuzzy inference model for a fuzzy control system. In fuzzy control systems, inference methods describe the algorithms for process control as a fuzzy relation between the information about the condition of the process and the control input of the process. Assume this fuzzy control system has two inputs \(x_1, x_2\), and an output \(y\), and there are \(M\) inference rules:

rule 1 \quad \text{IF } x_1 \text{ is } A_{11} \text{ and } x_2 \text{ is } A_{12} \text{ THEN } y \text{ is } B_1

rule 2 \quad \text{IF } x_1 \text{ is } A_{21} \text{ and } x_2 \text{ is } A_{22} \text{ THEN } y \text{ is } B_2

\ldots

rule M \quad \text{IF } x_1 \text{ is } A_{M1} \text{ and } x_2 \text{ is } A_{M2} \text{ THEN } y \text{ is } B_M

where \(A_{ii}, A_{i2},\) and \(B_i\) (\(i = 1 \text{ to } M\)) are fuzzy subsets.

When the inputs of the system are \(x_1 = x_1^o\) and \(x_2 = x_2^o\) (\(x_1^o\) and \(x_2^o\) are real numbers), the conclusion will be \(y = y^o\) (\(y^o\) is a real number). \(y^o\) will be given by the inference process which includes three steps. First, calculate the compatibility for each of the antecedent conditions as:

\[ c_i = A_{i1}(x_1^o) \cdot A_{i2}(x_2^o); \quad i = 1, 2, \ldots, M. \]
Then, the inference result for each rule is:

\[ y \text{ is } c_iB_i(y); \ i = 1, 2, \ldots, M. \]

The final inference result is constructed from \( c_iB_i(y) \) (\( i = 1, 2, \ldots, M \)):

\[ y \text{ is } B^*, \ B^*(y) = c_1B_1(y) \cup c_2B_2(y) \cup \cdots \cup c_MB_M(y) \]

The conclusion "\( y \text{ is } B^*\)" is a fuzzy conclusion. A non-fuzzy output for the control system \( y^o \) can be found by using a defuzzification method, such as set \( y^o \) equals to the central axis of the membership function of \( B^*(y) \):

\[ y^o = \frac{\int B^*(y)yd y}{\int B^*(y)d y} \quad (3.1) \]

From this example, we can see that a typical fuzzy-rule based system consists of four parts: a fuzzifier, a rule base, an inference engine, and a defuzzifier (see Fig. 1.3). The rule base is a set of fuzzy inference rules. In each inference rule, there is a premise and a consequence. The premise is
described by a fuzzy proposition and the consequence can be a fuzzy conclusion, or a linear model or some other model. A fuzzy inference rule for an N fuzzy inputs and P fuzzy outputs system can be expressed as:

\[
\text{IF: } \quad x_1 \text{ is } A_{i1}, \ x_2 \text{ is } A_{i2}, \ldots, \text{ and } x_N \text{ is } A_{iN} \\
\text{THEN: } \quad y_1 \text{ is } B_{i1}, \ y_2 \text{ is } B_{i2}, \ldots, \text{ and } y_M \text{ is } B_{ip}
\]

where \( X = \{x_n, \ n = 1, 2, \ldots, N\} \subset \mathbb{R}^N \) are the inputs to the fuzzy system, \( Y = \{y_p, \ p = 1, 2, \ldots, P\} \subset \mathbb{R}^P \) are the outputs, \( A_k \ (k = 1, 2, \ldots, N) \) and \( B_j \ (j = 1, 2, \ldots, P) \) are fuzzy subsets. Thus a fuzzy rule-based system implements a mapping from \( U \subset \mathbb{R}^N \) to \( V \subset \mathbb{R}^P \).

The fuzzification is a mapping from the observed input \( (x_i) \) to the fuzzy sets defined in the corresponding universe \( (A_k \text{ for } k = 1 \text{ to } N) \). The inference engine is a decision making logic algorithm which determines outputs corresponding to the fuzzified inputs with respect to the fuzzy inference rules. The designers must specify which implication, conjunction and aggregation operators are used. The defuzzification produces nonfuzzy outputs when necessary. Usually, three methods are used: Centre Of Area, Max Criterion, or Mean of Maximum [Terano et al., 1992].
The fuzzy inference rule base is the core of a fuzzy rule-based system. Fuzzification, the inference algorithm and defuzzification comprise the fuzzy inference process of a fuzzy rule-based system. There are many different choices within each of these four components, leading to many different models. In this research, fuzzy inference models for pattern recognition systems will be developed and implemented in neural network structures. The features of pattern recognition systems will be fully considered and the models should be suitable for neural network implementations.

A pattern recognition system is said to be supervised when the system is trained by a set of patterns with known classifications and is then asked to classify unknown patterns based on the information acquired during the training. Suppose that a pattern $X$ is represented in terms of $N$ features or properties $\{x_1, x_2, \ldots, x_N\}$, i.e., $X \in \mathbb{R}^N$. During the recognition procedure, a given pattern $X$ is to be assigned into one of the $P$ possible subsets or classes, $c_1, c_2, \ldots, c_P$ based on its feature values $\{x_1, x_2, \ldots, x_N\}$. Thus the recognition process is a mapping from $U \subseteq \mathbb{R}^N$ to $\{0,1\}^P$. If the subsets or classes $c_1, c_2, \ldots, c_P$ are fuzzy classes, then the system becomes a fuzzy classification system. The recognition process now is a mapping from $U \subseteq \mathbb{R}^N$ to the unit hypercube $[0,1]^P$. Because a fuzzy rule based system can be used to implement a mapping from $U \subseteq \mathbb{R}^N$ to $V \subseteq \mathbb{R}^n$, it also
can be used to implement a pattern recognition system, which is a mapping from 
\( \mathbb{U} \subset \mathbb{R}^n \) to \([0, 1]^p\).

Considering the features of pattern recognition systems, we use simplified 
inference rules for pattern recognition systems, in which the consequent parts are 
expressed in real numbers. In general, the inference rule set for a pattern 
recognition system is:

rule 1: IF \( x_1 \) is \( A_{11}, \ldots, \) and \( x_N \) is \( A_{N1}, \) THEN \( y_1 \) is \( y_{11}, \ldots, \) and \( y_P \) is \( y_{1P} \)

rule 2: IF \( x_1 \) is \( A_{12}, \ldots, \) and \( x_N \) is \( A_{N2}, \) THEN \( y_1 \) is \( y_{12}, \ldots, \) and \( y_P \) is \( y_{2P} \)

... 

rule \( M \): IF \( x_1 \) is \( A_{1M}, \ldots, \) and \( x_N \) is \( A_{NM}, \) THEN \( y_1 \) is \( y_{1M}, \ldots, \) and \( y_P \) is \( y_{MP} \)

where \( A_{ij}, A_{2i}, \ldots, A_{Nj} \) (\( j=1, 2, \ldots, M \)) are fuzzy subsets and \( y_{1j}, y_{2j}, \ldots, y_{pj} \) 
(\( j=1, 2, \ldots, M \)) are real numbers in [0, 1]. The membership functions of \( A_{ij} \) 
(\( i=1, 2, \ldots, N; j=1, 2, \ldots, M \)) can be any type of membership functions. For  
simplicity, only triangular functions are used in this research.

When the inputs of the system are \( x_1=x_1^o, \) and \( x_2=x_2^o, \ldots, x_N=x_N^o, \) the  
conclusion will be \( y_1=y_1^o, \) \( y_2=y_2^o, \ldots, y_P=y_P^o, \) where \( x_1^o, x_2^o, \ldots, x_N^o \) and \( y_1^o, \)
$y_2^a, \ldots, y_p^a$ are real numbers. Several inference algorithms are developed to obtain $y_1^a, y_2^a, \ldots, y_p^a$ from all the inference rules and the present inputs.

**Inference algorithm I, Min-Max inference:**

The compatibility for each of the antecedent conditions of the rules is

$$c_j = \text{MIN}_{i=1}^N (m_{ij}(x_i^a)) \quad j = 1, 2, \ldots, M$$  \hspace{1cm} (3.2)

where $m_{ij}(\cdot)$ is the membership function of fuzzy set $A_{ij}$. The complete inference results are

$$y_p^a = \text{MAX}_{j=1}^M (c_j y_{jp}) \quad p = 1, 2, \ldots, P$$  \hspace{1cm} (3.3)

**Inference algorithm II, Min-Sum inference:**

The compatibility for each of the antecedent conditions of the rules is
\[ c_j = \underbrace{\text{MIN}}_{i=1}^{N}(m_{ij}(x_i^o)) \quad j = 1, 2, ..., M \quad (3.4) \]

where \( m_{ij}(\cdot) \) is the membership function of fuzzy set \( A_{ij} \). The complete inference results are

\[
y_p^o = \begin{cases} 
\frac{\sum_{j=1}^{M} (c_j y_{jp})}{\sum_{j=1}^{M} c_j} & \text{if } \sum_{j=1}^{M} c_j \neq 0 \\
0 & \text{if } \sum_{j=1}^{M} c_j = 0 
\end{cases} \quad p = 1, 2, ..., P \quad (3.5) 
\]

**Inference algorithm III, Min-Competitive inference:**

The compatibility for each of the antecedent conditions of the rules is

\[ c_j = \underbrace{\text{MIN}}_{i=1}^{N}(m_{ij}(x_i^o)) \quad j = 1, 2, ..., M \quad (3.6) \]

where \( m_{ij}(\cdot) \) is the membership function of fuzzy set \( A_{ij} \). The complete inference
results are

\[ y_p^o = y_{jp} \quad \text{if} \quad c_j = t, \quad p = 1, 2, ..., P \quad (3.7) \]

\[ t = \max_{j=1}^{N} (c_j) \quad (3.8) \]

The fuzzification of these three inference models is implemented by calculating the membership values of each input to the fuzzy sets defined in the corresponding universe. In the inference process, minimum is used as conjunction operator, and three kinds of aggregation operators (weighted maximum, weighted sum and maximum competition) are used. Since the outputs of these three inference algorithms are real numbers in \([0, 1]\), defuzzification is not necessary. Decisions can be made based on the outputs, which express the membership values of the input pattern belonging to all the classes.

By using fuzzy neurons proposed in Chapter 2, the fuzzy inference models developed in this section will be implemented in neural network structures.
3.2 The Structures and Algorithms of Fuzzy Inference Networks

In this section, pattern recognition systems, which represent mappings from $U \subset \mathbb{R}^n$ to the unit hypercube $[0,1]^n$, will be implemented in neural network structures as distributed fuzzy rule-based systems by using the fuzzy neurons proposed in Chapter 2.

The proposed fuzzy inference networks are implemented in layered feedforward network structures. Different types of fuzzy neurons are used in different layers. Each layer has a different function from the other layers. The input layer accepts the data into the network. The output functions of the FNs in the first layer represent the membership functions of fuzzy subsets $A_{1j}$, $A_{2j}$, ..., $A_{nj}$ ($j=1, 2, ..., M$) in the IF parts of the inference rules. These output functions fuzzify the input data of the input patterns. The second layer represents inference rules. Each FN in the second layer computes the compatibility of the input pattern for one of the inference rules. The information in the THEN parts of the inference rules is stored in the weights from the FNs in the second layer to the FNs in the third layer. The third layer produces outputs from the information of the input pattern and all the inference rules.
Because three types of inference models are developed for pattern recognition systems in the previous section, three types of fuzzy inference networks, Min-Max Fuzzy Inference Network (MMFIN), Min-Sum Fuzzy Inference Network (MSFIN), and Min-Competitive Fuzzy Inference Network (MCFIN), will be constructed. Each of them is based on one of the inference methods.

TRAN-FNs are used in the first layers of all the three fuzzy inference networks. Each TRAN-FN accepts one feature of the input pattern and transfers it to M outputs (M is the number of FNs in the second layer). The \( i \)th TRAN-FN computes the membership values of the \( i \)th input with respect to the fuzzy sets defined in the \( i \)th projection of \( U \). So the output functions of the \( i \)th TRAN-FN in the first layer represent the membership functions of the fuzzy sets defined in the \( i \)th projection of \( U \). For simplicity, triangular functions are used in this research. The information in the IF parts of the inference rules is contained in these membership functions. The algorithm of the \( i \)th TRAN-FN is:

\[
y_j^1 = g_{y_j}(x_i), \quad j = 1, 2, \ldots, M
\]  

(3.9)

where \( g_y(x_i) \) is the \( j \)th output function of the \( i \)th TRAN-FN in the first layer.
$g_i(x_i)$ represents the membership functions of the fuzzy subset $A_i$. Three kinds of triangular functions as shown in Fig. 3.1 are used for the proposed fuzzy inference networks.

Fig. 3.1 Three types triangular membership functions
The second layer computes the compatibility for the antecedent condition of each fuzzy inference rule. MIN-FNs are used in the second layer to represent inference rules, one MIN-FN for each rule. Each MIN-FN has one output. The algorithm of the $j^{th}$ MIN-FN is:

$$y_j^2 = s_j^2 = \min_{i=1}^{N}(y_i^3)$$  \hspace{1cm} (3.10)

The information in the THEN parts of the inference rules is contained in the weights from the FNs in the second layer to the FNs in the third layer. The third layer gives the system outputs according to the fuzzy inference method. Because different inference methods were proposed, different types of fuzzy neurons are used in different types of fuzzy inference networks. For the Min-Max Fuzzy Inference Network (MMFIN), which is based on Min-Max inference method, MAX-FNs are used in the third layer (see Fig. 3.2).

The weighted inputs of the MAX-FNs contain the information of all the inference rules and the information of the input pattern. The algorithm of the $p^{th}$ MAX-FN is:
\[ y_p^3 = \max_{j=1}^M (w_{jp}^2 \cdot y_j^2) \] (3.11)

Fig. 3.2 Min-Max Fuzzy Inference Network

56
For the Min-Sum Fuzzy Inference network (MSFIN), SUM-FNs are used in the third layer, as shown in Fig. 3.3. Now the summations of the SUM-FNs combine the information of all the inference rules and the compatibilities of the input pattern. Then weighted averages are given as the inference outputs. The algorithm of the $p$th SUM-FN is:

\[
y_p^3 = \begin{cases} 
\frac{\sum_{j=1}^{M} (w_{jp}^2 \cdot y_j^2)}{\sum_{j=1}^{M} y_j^2} & \text{if } \sum_{j=1}^{M} y_j^2 \neq 0 \\
0 & \text{if } \sum_{j=1}^{M} y_j^2 = 0
\end{cases}
\]  (3.12)

For the Min-Competitive Fuzzy Inference Network (MCFIN) as shown in Fig. 3.4, COMP-FNs are used in the third layer. The threshold of all the COMP-FNs is the maximum of all the outputs from the FNs in the second layer. The algorithm of a COMP-FN is:

\[
y_p^3 = w_{jp}^2 \quad \text{if } y_j^2 = t^3, \quad p = 1, 2, \ldots, P
\]  (3.13)

57
\[ t^3 = \max_{j=1}^{M} (y_j^2) \]  \hspace{1cm} (3.14)

where \( t^3 \) is the threshold of all the COMP-FNs.

Fig. 3.3 A Min-Sum Fuzzy Inference Network
Fig. 3.4 A Min-Competitive Fuzzy Inference Network
Each layer of the proposed FINs has a different function. The input layer fuzzifies the input pattern and gives the membership values of the input data with respect to the fuzzy sets defined in each projection of U. The second layer represents inference rules and computes the compatibility for the antecedent condition of each fuzzy inference rule. The third layer combines the information of all the inference rules and the information of the input pattern and then gives outputs according to the inference method.

In this section, the proposed fuzzy inference models for pattern recognition systems are implemented in neural network structures using fuzzy neurons. It is obvious that the key point of implementing a fuzzy inference network as a pattern recognition system is to determine the fuzzy inference rules from the training data, i.e., to determine the number of inference rules, the membership functions of the fuzzy subsets in the antecedent part and the outputs in the conclusion part in each rule. In the next section, the learning capability of the proposed FINs and the learning strategy for determining the inference rules from training data will be discussed.
3.3 The Learning Strategy of Fuzzy Inference Networks

In the previous section, three types of fuzzy inference networks have been developed. From chapter 2, we know that the parameters or functions that describe the interactions among fuzzy neurons, such as the weights and the output functions can be adjusted during the learning procedure. Because all the information of the inference rules is contained in the weights and output functions of the fuzzy neurons in the networks, it becomes possible to determine the inference rules from the training data by adjusting the weights and output functions of the fuzzy neurons during the training procedure.

In order to provide a fuzzy inference system with learning capability, various learning methods have been proposed. The most widely used method is the descent method (backpropagation learning), which can optimize the shape of membership functions in inference rules from input-output training data. However, this method has suffered from the inherent prerequisite problems, such as an advanced setting of the number of inference rules, which have to be derived by trial and error [Higgins and Goodman, 1994]. The learning results depend on the number of inference rules. If the number of inference rules is small, the
system cannot express the input-output relation properly. If the number is large, the system will be too expensive and the generalization capability is not good. Therefore, the number of inference rules has to be determined from an overall point of view. The work to determine the number of inference rules requires a designer with a lot of experience. In order to solve this problem, a new type of learning method for fuzzy inference networks, a self-organizing learning algorithm, is proposed in this research. In this kind of learning algorithm, the number of inference rules, the membership functions in the antecedent parts, and the outputs in the conclusion parts of inference rules will be automatically determined during the training procedure.

The training set for a supervised pattern recognition system is in the form of \( \{X_k, Y_k \mid k=1, \ldots, L\} \), where \( X_k = \{x_{k1}, \ldots, x_{kN}\} \subset \mathbb{R}^N \) and \( Y_k = \{y_{k1}, \ldots, y_{kp}\} \subset [0,1]^p \). They could be expressed in the form of "IF THEN" rules, that is, \{ IF \( x_i \) is \( x_{ki} \), \ldots, and \( x_N \) is \( x_{kN} \), THEN \( y_1 \) is \( y_{k1} \), \ldots, and \( y_p \) is \( y_{kp} \) \mid \( k=1, 2, \ldots, L \} \). These "IF THEN" rules are similar to fuzzy inference rules. If the non-fuzzy values in the IF parts are fuzzified, these "IF THEN" rules can be used in the proposed fuzzy inference networks. In the fuzzification procedure, the non-fuzzy value \( x_{ki} \) is associated with a fuzzy number \( A_{ki} (i=1, 2, \ldots, N) \). Then these "IF THEN" rules become: \{ IF \( x_i \) is \( A_{ki} \), \ldots, and \( x_N \) is \( A_{kN} \), THEN \( y_1 \) is \( y_{k1} \), \ldots, and
$y_p$ is $y_{kp} \mid k=1, 2, \ldots, L$, which are the same as the inference rules of the proposed fuzzy inference models for pattern recognition systems. Consequently, the training procedure of the proposed fuzzy inference networks becomes a procedure of selecting prototypes from the training patterns and doing some fuzzification. The fuzzification will be performed by adjusting the parameters of the output functions of TRAN-FNs in the first layer. Inference rules will be established based on the prototypes selected from the training set, one prototype for each rule. The MIN-FNs in the second layer represent inference rules, so they will represent the prototypes. The number of MIN-FNs will be determined by the number of prototypes. The information about the conclusion parts of the inference rules is contained in the weights from the FNs in the second layer to the FNs in the third layer. So these weights will be determined by the desired outputs of the prototypes.

In the learning algorithms, the fuzzy inference rules will be established in such a way that the resultant networks can recall all the training patterns within the error tolerance $E_t$, i.e., the criterion of the learning algorithms for the fuzzy inference networks is that the maximum estimated error for all the training samples should be less than or equal to the error tolerance $E_t$. This is different from those backpropagation learning algorithms, which adjust parameters to
minimize the sum of the square of estimated errors. In backpropagation learning algorithms, the training procedure has to be performed for many epochs in order to achieve the optimal results. However, the training procedure of the fuzzy inference networks will finish in a few epochs. So the training of the fuzzy inference networks is much faster than those networks using backpropagation learning, in addition to having the advantage of determining inference rules automatically.

The objectives of the learning procedure for the fuzzy inference networks are to determine the inference rules by establishing MIN-FNs in the second layer and to adjust the parameters of the FNs in the input and the third layers according to the training data. Prototypes will be selected from the training samples and then they will be fuzzified to establish an inference rule. If a pattern $X_k = \{x_{1k}, x_{2k}, \ldots, x_{nk}\}$ with desired outputs $\{y_{k1}, y_{k2}, \ldots, y_{kp}\}$ is selected as a prototype, then an inference rule will be established as:

Rule j: 

\[
\text{IF } m_{1k}(x_{1k}) = 1.0, m_{2k}(x_{2k}) = 1.0, \ldots, \text{ and } m_{nk}(x_{nk}) = 1.0,
\]

\[
\text{THEN } y_1 \text{ is } y_{k1}, \ y_2 \text{ is } y_{k2}, \ldots, \text{ and } y_p \text{ is } y_{kp}.
\]

where $m_{ik}(\ )$ is the membership function for fuzzy number $A_{ik}$, and $y_{kp}$
(p = 1, 2, ..., P) are the desired outputs for the pattern \( \mathbf{X}_k \). The training data set and the way of selecting prototypes will determine the inference rules.

The output functions of the TRAN-FNs in the first layer fuzzify the feature data of input patterns. They represent the membership functions of the fuzzy subsets in the IF parts of the inference rules. Three types of triangular functions are used in this research: triangular functions with different threshold in each side, non-isosceles triangular functions, and isosceles triangular functions (as shown in Fig. 3.1). The parameters of the membership functions will be determined according to the input data of the selected prototypes.

The weights from the MIN-FNs in the second layer to the FNs in the third layer can be determined by utilizing Hebb's rule through learning algorithms. Hebb's rule can be expressed as:

When unit A and unit B are simultaneously excited, increase the strength of the connection between them.

When a MIN-FN in the second layer is established during the learning, it is treated as being excited. All the FNs in the third layer can be treated as being
excited to some degree according to the desired outputs. So the weights between
the established MIN-FN in the second layer and the FNs in the third layer can be
determined according to the desired outputs of the training data.

There are five steps in the training procedure:

**Step 1:** Set the error tolerance \( E_i \).

**Step 2:** Determine the shape of the membership functions in the antecedent
part of each inference rule, that is to determine the shape of \( g_i(x_i) \)
in equation 3.9.

**Step 3:** Select prototypes from the training patterns. Establish MIN-FNs
in the second layer to represent inference rules based on the
selected prototypes.

**Step 4:** Adjust the parameters of the membership functions and the weights
from the second layer to the third layer according to the selected
prototypes.

**Step 5:** Input all the training pattern to the network and compute the output
error as:
\[ e = \max_{p=1}^{P} |d_p - y_{p}^{3}| \]  

(3.15)

where \( d_p \) is the desired output and \( y_{p}^{3} \) is the actual output. If \( |e| \leq E_t \), then stop learning. Otherwise, add MIN-FNs in the second layer for those training patterns that do not satisfy equation (3.15). Repeat Step 5 until all the \( |e| \leq E_t \).

Different methods are used in this research for determining inference rules, selecting prototypes from training patterns and determining the structures of the networks. Details will be discussed in the next two chapters.
3.4 Capabilities of Fuzzy Inference Networks

3.4.1. Functions of each layer

The input layers of the proposed fuzzy inference networks fuzzify input patterns. Fuzzification is performed by choosing triangular membership functions as the output functions of the TRAN-FNs in the input layer. These triangular functions give the membership values of the input features \( x_i \) (\( i = 1, 2, \ldots, N \)) with respect to the fuzzy subsets \( A_{ij} \) (\( i = 1, 2, \ldots, N, j = 1, 2, \ldots, M \)) in the inference rules.

The second layers compute the distances or the similarities of the input patterns to the prototypes. There are many methods to compute the distances between patterns, such as the dot products, the Euclidean distances, the Hamming distances, and the fuzzy similarities. Since fuzzy methods are used in the proposed fuzzy inference networks, fuzzy similarities are used as the distances in this research.

The third layers determine the outputs according to the inference rules and corresponding inference methods. Because different fuzzy inference methods are
employed for different fuzzy inference networks, different FNs are used in the third layer.

The proposed fuzzy inference networks are distributed fuzzy rule-based systems. The fuzzy inference rules are stored in the connections between neurons in the networks. The fuzzy neurons fuzzify input patterns, do fuzzy reasoning and then defuzzify the outputs. All the rules are fired at the same time to different degrees and the networks can give multiple outputs simultaneously. Consequently, the proposed fuzzy inference networks are distributed fuzzy rule-based systems with learning ability.

The proposed fuzzy inference network pattern recognition systems implement mappings from \( U \subset \mathbb{R}^N \) to \([0,1]^p\). It is easy to decompose the input-output mapping into three mappings, one for each layer of the networks, as \( \Phi = \Phi_2 \circ \Phi_1 \circ \Phi_0 \), where \( \Phi_0 \) is a mapping from \( U \) to \([0,1]^{N \times M} \), \( \Phi_1 \) is a mapping from \([0,1]^{N \times M} \) to \([0,1]^M \), and \( \Phi_2 \) is a mapping from \([0,1]^M \) to \([0,1]^p \). The parameters of \( \Phi_0 \) define the membership functions of the fuzzy subsets in the IF parts of the inference rules. \( \Phi_1 \) has no adjustable parameter if the number of inference rules is determined. The parameters of \( \Phi_2 \) determine the THEN parts of the inference rules.
3.4.2. Learnability

The proposed fuzzy inference networks are composed of fuzzy neurons. Fuzzy neurons can learn from environments. The weights, the output functions and the activation thresholds can be adjusted during the learning procedure. So the fuzzy inference networks have learning ability. The learning procedure of the proposed fuzzy inference networks will determine the inference rules from the training data and establish a network that can recall all the training patterns within an error tolerance $E_i$.

3.4.3. Plasticity

As learning proceeds, the fuzzy inference networks are generated adaptively in response to a series of environmental inputs and outputs. When the system discovers and learns a prototype, a MIN-FN is used to represent the prototype and the interactions between this MIN-FN and the other FNs in the input and output layers are adjusted. The structures and the performances of the fuzzy inference networks are determined by the training sets.
3.4.4. Stability

The proposed fuzzy inference networks are feedforward networks. There is no feedback between fuzzy neurons in the networks. Consequently, the fuzzy inference networks are stable systems.

3.4.5. Convergence of the learning

The learning of the proposed fuzzy inference networks is a procedure of selecting prototypes from training data and determining inference rules. Since the prototypes will be selected in such a way that the resultant networks can recall all the training patterns within the error tolerance $E_0$, the learning is always convergent because the extreme case is to select all the training patterns as prototypes. The way of selecting prototypes and the training data set will affect learning speed but our learning experience shows that it will finish in several epochs.
3.4.6. Realizability

Five types of fuzzy neurons are used in the proposed fuzzy inference networks. The operations of minimum, maximum, summation, competition, and product are employed in these fuzzy neurons. These operations can be easily realized by present techniques. It is difficult to realize complex and irregular membership functions. However, triangular membership functions can be realized easily by taking the minimum operation of two linear functions. The output functions of the TRAN-FNs are triangular membership functions. Consequently, the proposed fuzzy inference networks are realizable.

3.4.7. Complexity

The proposed fuzzy neurons employ not only product and summation but also minimum, maximum, competition and nonlinear output functions. Kohonen (Kohonen, 1982) points out that the computations of minimum and maximum are simpler than that of the product. The competition function of COMP-FNs can be implemented by switches. The nonlinear output functions are more difficult to
be implemented than the product. However, if trapezoidal functions are chosen as the output functions, they can be implemented by taking the minimum operation of three linear functions. In addition, the proposed fuzzy inference networks have avoided backpropagation-based learning algorithms, which require excessive computational hardware for on-chip learning. So a system composed of the fuzzy neurons with simple output functions will not be more complex than a neural network system composed of traditional neurons.

The structures of the proposed fuzzy inference networks are very simple. They are layered feedforward networks composed of fuzzy neurons. Each layer has its own function. The complexity of the proposed fuzzy inference networks is determined by the output functions of the TRAN-FNs in the first layer. The size of the fuzzy inference networks is determined by the training set and the way of establishing fuzzy inference rules. The numbers of FNs in the input and the output layer are determined by the training sets. The number of FNs in the second layer is determined by both the training set and the learning algorithm.
3.4.8. Potential of fuzzy inference networks

When used for practical problems, the proposed fuzzy inference networks may need more than one fuzzification layer to fuzzify several types of features of the input patterns. For example, a visual pattern recognition system may need two different fuzzification layers for the grey values and the position information of important pixels. Two such fuzzy inference networks will be developed in Chapter 5 for letter recognition systems.

For complicated problems, several basic fuzzy inference networks can be used to form a multi-level fuzzy inference system. For example, a three level fuzzy inference system can be used as a word recognition system. The first level is the segment recognition level, the second is the letter recognition level, and the third is the word recognition level. Each level is composed of several fuzzy inference networks. In this way, large intelligent systems that utilize fuzzy inference methods can be constructed by using fuzzy inference networks as their sub-systems. Due to the time limitation, the research on multi-level fuzzy inference systems is not included in this dissertation. However, it is a direction suggested for further fuzzy inference network research.
CHAPTER IV

FUZZY INFEERENCE NETWORKS FOR PATTERN CLASSIFICATION

4.1 Features of Fuzzy Classifiers

Classification of objects is an important area of research and of practical applications in a variety of fields, including pattern recognition, artificial intelligence, and vision analysis. In a pattern recognition problem, if the \textit{a priori} probabilities and the state conditional densities of all the classes are known, Bayes decision theory produces optimal results in the sense that it minimizes the expected error rate [Keller, Gray and Givens, 1985]. However, in many pattern recognition problems, this information is not available. In this case, many other algorithms such as nearest prototype algorithm, K-nearest neighbour (K-NN) algorithm, and neural network classification algorithms are used.

In a pattern classification problem, a given pattern $X$ is to be assigned into one of the $M$ possible subsets or categories (classes), $c_1, c_2, \ldots, c_M$, based on its feature values $\{x_1, x_2, \ldots, x_N\}$. A pattern $X=\{x_1, x_2, \ldots, x_N\}$ can be viewed as
a point in a N-dimensional real number space \( \mathbb{R}^N \). Conventional non-fuzzy or crisp classification techniques assume that a pattern \( X \) can belong to one and only one class. Fuzzy classification algorithms assign the pattern \( X \) with a distributed membership value to each class, which is to assign fuzzy labels to the pattern \( X \). The partitions between fuzzy classes are 'soft'. When there is some overlap between the classes, fuzzy subsets offer special advantages over conventional non-fuzzy classification methods. The fuzzy labels of a fuzzy classifier can be "defuzzified" and then the fuzzy classifier becomes a hard classifier, but uses the idea of fuzziness in the model. Fuzzy classifier design almost always means arriving at a hard classifier because most pattern recognition systems require hard labels for objects being classified. We can find a better solution to a crisp problem by looking in a larger space at first, which gives the algorithm more freedom to avoid more errors.

Fuzzy classifier design can be performed by supervised learning using a set of training data with fuzzy or non-fuzzy labels. After being designed, it is asked to classify unlabelled patterns. When given a pattern, the fuzzy classifier computes the membership values of the pattern in each class and makes decisions based on these membership values. Since 1965, many efforts have been dedicated to fuzzy classification and many algorithms have been presented and applied in
pattern recognition and decision systems. Among all the work that has been done, the fuzzy K-nearest neighbour algorithm by Keller et al. [Keller et al., 1985] and the fuzzy c-means algorithm by Bezdek [Bezdek, 1981] are the most important ones for pattern recognition problems. Fuzzy classification algorithms have advantages when the patterns have overlap characteristics, or a decision maker needs the information of classification uncertainty, or the features of patterns have some uncertainty, or it is difficult to find a hard boundary in a classification problem.

In this chapter, Min-Max and Min-Sum Fuzzy Inference Networks are used as classifiers. After being trained by labelled data, the proposed fuzzy inference networks can find the fuzzy and hard partitions between the classes. Efficient self-organizing learning algorithms are also developed. The fuzzy inference network classifiers build decision boundaries by creating subsets of the pattern space. They are model-free estimators and do not make assumptions of how outputs depend on inputs. Instead, they adjust themselves to a given training set and decide the boundaries of classes. When given an unknown pattern, the fuzzy inference network classifiers use the learned knowledge to estimate the membership values of this pattern in each class and classify the input pattern according to the membership values.
4.2. Min-Max and Min-Sum Fuzzy Inference Classification Algorithms

In Chapter 3, fuzzy inference networks have been developed as pattern recognition systems, which represent mappings from the pattern space $U \subset \mathbb{R}^N$ to the unit hypercube $[0,1]^N$. The proposed fuzzy inference network pattern recognition systems can be expressed as a set of fuzzy inference rules and a specified inference algorithm. The fuzzy inference rules are in the form of:

rule 1: IF $x_1$ is $A_{11}$, ..., and $x_N$ is $A_{Ni}$, THEN $y_1$ is $y_{11}$, ..., and $y_p$ is $y_{1p}$

rule 2: IF $x_1$ is $A_{12}$, ..., and $x_N$ is $A_{N2}$, THEN $y_1$ is $y_{12}$, ..., and $y_p$ is $y_{2p}$

... 

rule M: IF $x_1$ is $A_{1M}$, ..., and $x_N$ is $A_{NM}$, THEN $y_1$ is $y_{1M}$, ..., and $y_p$ is $y_{MP}$

where $A_{ij}$, $A_{2j}$, ..., $A_{Nj}$ ($j=1, 2, ..., M$) are fuzzy subsets and $y_{j1}$, $y_{j2}$, ..., $y_{jp}$ ($j=1, 2, ..., M$) are real numbers in $[0,1]$.

When triangular functions are used as the membership functions of $A_{ij}$ ($i=1, 2, ..., N$) and "MIN" are used as the fuzzy "AND" operation, the IF part of the $j$th rule actually represents a fuzzy hyperbox in the pattern space $U \subset \mathbb{R}^N$. 

78
A fuzzy hyperbox is a fuzzy set defined on a hyperbox. Fig 4.1 illustrates a three-dimensional hyperbox. The prototype of the $j^{th}$ rule, $\{x_{i1}, x_{i2}, \ldots, x_{in}\}$, is the key point of the fuzzy hyperbox. The membership function of the hyperbox is determined by the membership functions $m_{ij}(\cdot)$ of fuzzy sets $A_i$ ($i=1, 2, \ldots, N$).

The membership function of the $j^{th}$ fuzzy hyperbox is:

$$c_j(X) = c_j(x_1, x_2, \ldots, x_n) = \text{MIN}[m_{i1}(x_1), m_{i2}(x_2), \ldots, m_{in}(x_n)]$$

Each output of the fuzzy inference network describes the membership value of the input pattern belonging to one fuzzy class. The network deduces outputs according to the inference rules and the specified inference algorithm. Each fuzzy class is an aggregation of the fuzzy hyperboxes. Assuming $B_j$ represents the $j^{th}$ hyperbox ($j=1, 2, \ldots, M$), $c_j(X)$ is the membership function of $B_j$, then each fuzzy $F_p$ ($p=1$ to $P$) class is an aggregation of $B_j$:

$$F_p = \bigcup_{1 \leq j \leq M} B_j$$

The union operation is weighted MAX for Min-Max inference method and is weighted SUM for Min-Sum inference method, i.e., for Min-Max inference,
hyperbox

centre \( (x_1, x_2, x_3) \)

Fig 4.1 A three-dimensional hyperbox
\[ y_p = \max_{j=1}^{M} (c_j \cdot y_{jp}) \quad p = 1, 2, \ldots, P \] (4.1)

and for Min-Sum Inference,

\[
y_p = \begin{cases} 
\sum_{j=1}^{M} (c_j \cdot y_{jp}) & \text{if } \sum_{j=1}^{M} c_j \neq 0 \\
\sum_{j=1}^{M} c_j & \text{if } \sum_{j=1}^{M} c_j = 0
\end{cases} \quad p = 1, 2, \ldots, P \] (4.2)

where \( c_j \) is the membership value of the input pattern belonging to the \( j \)th fuzzy hyperbox \( B_p \), \( y_{jp} \) is the membership value of the \( j \)th prototype belonging to the \( p \)th class, and \( y_p \) is the membership value of the input pattern belonging to the \( p \)th class \( F_p \). For nonfuzzy pattern classification, \( y_{jp} \) is 0 or 1, then the outputs for Min-Max Inference become:

\[ y_p = \max_{y_{jp}=0} (c_j) \quad p = 1, 2, \ldots, P \] (4.3)
and for Min-Sum Inference,

$$\begin{align*}
\begin{cases}
\sum_{j=1}^{M} c_j & \text{if } \sum_{j=1}^{M} c_j \neq 0 \\
0 & \text{if } \sum_{j=1}^{M} c_j = 0
\end{cases}
\end{align*}
$$

(4.4)

$$p=1,2,\ldots,P$$

From equation (4.1) to equation (4.4), we note that the membership value of the input pattern belonging to the $p$th class $F_p$ is determined by the most important prototype for Min-Max inference method and by all the important prototypes for Min-Sum inference method. The "most important prototype" for an input pattern regarding the $p$th class is defined as $c_j y_{jp} = \max(c_j y_{jp})$. And an "important prototype" for an input pattern regarding the $p$th class is defined as $c_j \neq 0$ and $y_{jp} \neq 0$.

Let $W = \{ X_1, X_2, \ldots, X_M \}$ be the set of prototypes, where $X_j = \{ x_{1j}, x_{2j}, \ldots, x_{Nj} \} \in \mathbb{R}^N$. The classification algorithms are:
Min-Max Fuzzy Inference Classification Algorithm

BEGIN

input pattern \( X = \{ x_1, x_2, \ldots, x_N \} \) of unknown classification

initialize \( j = 1 \)

\textbf{DO UNTIL} ( \( j > M \))

compute the membership values of \( X \) to all the hyperboxes as

\[
c_j = \min_{i=1}^{N} m_{ij}(x_j)
\]

where \( m_{ij}(\cdot) \) is the membership function of fuzzy set \( A_{ij} \)

increment \( j \)

\textbf{END DO UNTIL}

initialize \( p = 1 \)

\textbf{DO UNTIL} ( \( p > P \))

calculate the membership values of \( X \) to the fuzzy classes \( F_i \) by finding the most important prototype as

\[
\gamma_p = \max_{j=1}^{M} (c_j \cdot \gamma_{jp})
\]
where $y_{ip}$ is the membership value of the prototype $X_j$ to class $F_p$

increment $p$

END DO UNTIL

END

Min-Sum Fuzzy Inference Classification Algorithm

BEGIN

input pattern $X = \{x_1, x_2, \ldots, x_N\}$ of unknown classification

initialize $j=1$

DO UNTIL ($j > M$)

compute the membership values of $X$ to all the hyperboxes as

$$c_j = \min_{i=1}^{N} m_{ij}(x_j)$$

where $m_{ij}(\cdot)$ is the membership function of fuzzy set $A_{ij}$

increment $j$

END DO UNTIL
initialize p = 1

DO UNTIL ( p > P )

calculate the membership values of X to the fuzzy classes F_p by finding the important prototype as

\[
y_p = \begin{cases} 
\frac{\sum_{j=1}^{M} (c_j \cdot y_{j,p})}{\sum_{j=1}^{M} c_j} & \text{if } \sum_{j=1}^{M} c_j \neq 0 \\
0 & \text{if } \sum_{j=1}^{M} c_j = 0
\end{cases}
\]

where \( y_{j,p} \) is the membership value of the prototype \( X_j \) to class \( F_p \)

increment p

END DO UNTIL

END

In nonfuzzy classification cases, the input pattern will be assigned to the class with the largest membership value or to the last found class with the largest membership value.
4.3. Network Structures and Learning Algorithms of FIN Classifiers

4.2.1. Network structures of FIN classifiers

From the previous discussion, we can see that Min-Max Fuzzy Inference Network (MMFIN) and Min-Sum Fuzzy Inference Network (MSFIN) can be used as classifiers.

The structure of a two-dimensional MMFIN classifier is shown in Fig. 4.2. The first layer accepts an input pattern and transfers the features of the pattern into membership values. We use TRAN-FNs in this layer. There are N TRAN-FNs in the first layer for a N-dimensional classifier. Each TRAN-FN has one input and M outputs (M is the number of FNs in the second layer). The outputs of the TRAN-FNs are:

$$
y^i_j = g_j(x_i) = \begin{cases} 
1 + \alpha (x_i - \theta_{ij}) & \text{if } 1 \geq 1 + \alpha (x_i - \theta_{ij}) > \epsilon^1_{ij} \\
1 - \alpha (x_i - \theta_{ij}) & \text{if } 1 \geq 1 - \alpha (x_i - \theta_{ij}) > \epsilon^2_{ij} \\
0 & \text{otherwise}
\end{cases} \quad (4.5)
$$

$$
i = 1, 2, \ldots, N; \quad j = 1, 2, \ldots, M$$
where $x_i$ is the input of the $i$th FN which represents the $i$th feature value of an input pattern. $g_{ij}(x_i)$ is the $j$th output function of the $i$th TRAN-FN. $g_{ij}(x_i)$ is a triangular function with thresholds and is shown in Fig. 4.3. The threshold $t_{ij}^1$ and $t_{ij}^2$ are of positive values. Parameters $\Theta_{ij}$, $\alpha$, and $t_{ij}^1$, $t_{ij}^2$ are to be determined by the learning algorithm.

Fig. 4.2 Min-Max Fuzzy Inference Network as a classifier
The second layer consists of MIN-FNs. The number of MIN-FNs, M, is to be determined by the learning algorithm. Each MIN-FN in this layer has N inputs and one output:
\[ y_j^2 = \min_{i=1}^{N} (y_{ij}^1) \quad j=1, 2, \ldots, M \quad (4.6) \]

There are \( P \) MAX-FNs in the third layer of the MMFIN classifier (\( P \) is the number of the classes). The algorithm is:

\[ y_p^3 = \max_{j=1}^{M} (w_{jp}^2 \cdot y_j^2) \quad p=1, 2, \ldots, P \quad (4.7) \]

where \( w_{jp}^2 \) is the connection weight between the \( j \)th MIN-FN in the second layer and the \( p \)th MAX-FN in the third layer and is to be determined by the learning algorithm. \( y_p^3 \) gives the membership value of the input pattern to the \( p \)th fuzzy class.

A two-dimensional Min-Sum Fuzzy Inference Network classifier is shown in Fig. 4.4. As in the MMFIN classifier, TRAN-FNs are used in the first layer and MIN-FNs are used in the second layer of the MSFIN classifier. However, the output functions of the TRAN-FNs in the MSFIN classifier are different from those of the MMFIN classifier. They are triangular functions with different slopes on each side (see Fig. 4.5).
Fig. 4.4 Min-Sum Fuzzy Inference Network as a classifier
Fig. 4.5 Output function of TRAN-FNs in the MSFIN classifier
The algorithms for the first and second layers of MSFIN classifier are:

\[
y^1_{ij} = g^1_{ij}(x_i) = \begin{cases} 
1 + \alpha^1_{ij}(x_i - \theta_{ij}) & \text{if } 1 \geq 1 + \alpha^1_{ij}(x_i - \theta_{ij}) \geq 0 \\
1 - \alpha^2_{ij}(x_i - \theta_{ij}) & \text{if } 1 \geq 1 - \alpha^2_{ij}(x_i - \theta_{ij}) \geq 0 \\
0 & \text{otherwise}
\end{cases} \\
i = 1, 2, \ldots, N; \quad j = 1, 2, \ldots, M
\]

(4.8)

\[
y^2_j = \min_{i=1}^{N} (y^1_{ij}) \\
j = 1, 2, \ldots, M
\]

(4.9)

where \(\alpha^1_{ij}, \alpha^2_{ij}\) and \(M\) are to be determined by the learning algorithm.

SUM-FNs are used in the third layer of the MSFIN classifier. Each SUM-FN represents one fuzzy class. The algorithm is:

\[
y^3_p = \begin{cases} 
\frac{\sum_{j=1}^{M} (w^2_{jp} \cdot y^2_j)}{\sum_{j=1}^{M} y^2_j} & \text{if } \sum_{j=1}^{M} y^2_j \neq 0 \\
0 & \text{if } \sum_{j=1}^{M} y^2_j = 0
\end{cases} \\
p = 1, 2, \ldots, P
\]

(4.10)
After being trained by labelled data, the proposed fuzzy inference network classifiers can find fuzzy or nonfuzzy partitions between classes. When given a pattern, the fuzzy inference network classifiers estimate the membership values of this pattern belonging to all the classes and make decisions based on these membership values. Since most pattern recognition systems require nonfuzzy labels for patterns being classified, the fuzzy labels of the fuzzy inference network classifiers should be defuzzified in this case. The maximum membership rule can be employed to defuzzify the fuzzy labels. The maximum membership rule is expressed as:

Assuming there are P fuzzy sets: $F^1$, $F^2$, ..., $F^p$. For a given pattern $X$, if $1 \leq i \leq P$ and:

$$M_i(X) = \max_{j=1}^{P} [M_j(X)]$$  \hspace{1cm} (4.11)

where $M_i(X)$ is the membership value of the pattern $X$ related to $F^i$ ($j=1$ to $P$), then $X$ most likely belongs to $F$. If more than one maximum membership values exist, $X$ is assigned to the last found class with the maximum membership value.
4.3.2. Learning algorithms of FIN classifiers

In this section, self-organizing learning algorithms are developed for the proposed fuzzy inference network classifiers. During the learning procedure, the fuzzy inference rules and the fuzzy inference networks are automatically established to match all the training samples within the error tolerance.

The parameters of the output functions of the TRAN-FNs in the first layer, the number of MIN-FNs in the second layer, and the weights between the second layer and the third layer have to be determined during the training procedure.

Before describing the algorithm, we define $c_{kp}$ as the desired $p$th output of the $k$th training sample ($p = 1$ to $P$ and $k = 1$ to $L$) and $D$ as the largest distance between any two training patterns.
Self-organizing learning algorithm for the MMFIN classifier:

BEGIN

set $E_0$, set $\alpha = 1/D$

establish layer 1 and layer 3

establish the 1st MIN-FN in layer 2 for the 1st training sample

initialize the output functions $g_{ni}(\ )$ and weights $w_{2p}$ by setting $\Theta_{ni} = x_{ni}$,

$t_{ni}^1 = 0$, $t_{ni}^2 = 0$ (i=1 to N), and $w_{2p} = d_{1p}$ (p=1 to P)

initialize m=1, k=2

DO UNTIL (k $\geq$ K)

input the kth training pattern, compute the output errors:

$e_p = y_p^2 - d_p$, (p=1 to P)

IF ( there is one $e_p > E_i$ (p=1 to P) ), THEN

DO UNTIL ( $e_p \leq E_i$ (p=1 to P) )

adjust $t_{ni}^1$, and $t_{ij}^2$ (i=1 to N, j=1 to m) according to the following rule:

IF $y_p^2 = w_{2p}^j y_j^2$ and if $| x_{nk} - \Theta_{ij} | = \text{MAX} | x_{nk} - \Theta_{ij} |$, THEN $1 \leq i \leq N$

then set $t_{ij}^1 = y_j^2$ if $x_{nk} < \Theta_{ij}$, or set $t_{ij}^2 = y_j^2$ if $x_{nk} > \Theta_{ij}$

input the kth training pattern, compute the output errors

END IF

95
END DO UNTIL

END IF

IF (there is one \( e_p < -E_i \) (\( p=1 \) to \( P \)) ) THEN

add a new MIN-FN in layer 2 and increment \( m \)

initialize the output functions \( g_{im}(\cdot) \) and weights \( w_{im}^2 \) by

setting \( \Theta_{im} = x_{ik} \), \( t_{im}^1 = 0 \), \( t_{im}^2 = 0 \) (\( i=1 \) to \( N \)), and \( w_{im}^2 = d_{ip} \)

(for \( p=1 \) to \( P \))

END IF

increment \( k \)

END DO UNTIL

DO UNTIL (the largest output error \( \sigma < E_j \))

initialize \( k=1 \)

DO UNTIL (\( k \geq K \))

input the \( kth \) training samples and compute the output error

IF (there is one \( e_p > E_i \) (\( p=1 \) to \( P \)) \( \sigma > E_i \)), THEN

DO UNTIL (\( e_p > E_i \) (\( p=1 \) to \( P \)) )

adjust \( t_{ij}^1 \) and \( t_{ij}^2 \) (\( i=1 \) to \( N \), \( j=1 \) to \( m \)) according to the following rule:

\[ IF \ y_p^3 = w_{jp}^3 y_j^3 \ \text{and if} \ \max_{1 \leq i \leq N} | x_{ik} - \Theta_{ij} | = \max_{1 \leq i \leq N} | x_{ik} - \Theta_{ij} |, THEN \]

96
set \( t_{ij}^1 = y_j^1 \) if \( x_{ik} < \Theta_{ij} \), or set \( t_{ij}^2 = y_j^2 \) if \( x_{ik} > \Theta_{ij} \)

input the \( kth \) training pattern, compute the output errors

END IF

END DO UNTIL

END IF

IF ( there is one \( e_p < -E_i \) (\( p = 1 \) to \( P \)) ) THEN

add a new MIN-FN in layer 2 and increment \( m \)

initialize the output functions \( g_{im} \) and weights \( w_{im}^2 \) by setting \( \Theta_{im} = x_{ik} \), \( t_{im}^1 = 0 \), \( t_{im}^2 = 0 \) (\( i = 1 \) to \( N \)), and \( w_{im}^2 = j_{kp} \)

(for \( p = 1 \) to \( P \))

END IF

\[
\sigma_k = \max_p (|e_p|)
\]

increment \( k \)

END DO UNTIL

\[
\sigma = \max_k (\sigma_k)
\]

END DO UNTIL

END
Self-organizing learning algorithm for the MSFIN classifier:

BEGIN

set \( F_1 \), establish layer 1 and layer 3

establish the 1st MIN-FN in layer 2 for the 1st training sample

initialize the output functions \( g_{ij}( ) \) and weights \( w_{ip}^2 \) by setting \( \Theta_{ii} = x_{ii} \),
\( \alpha_{ii}^1 = 1/D, \alpha_{ii}^2 = 1/D (i = 1 \text{ to } N) \), and \( w_{ip}^2 = d_{ip} (p = 1 \text{ to } P) \).

initialize \( m = 1, k = 2 \)

DO UNTIL \( (k \geq K) \)

input the \( k/h \) training pattern, compute the output errors:

\[ e_p = y_p^3 - d_{kp}, \quad (p = 1 \text{ to } P) \]

IF ( there is one \( | e_p | > E \), \( (p = 1 \text{ to } P) \)), THEN

add a new MIN-FN in layer 2 and increment \( m \)

set \( \Theta_{im} = x_{ik}, \alpha_{im}^1 = 1/D, \alpha_{im}^2 = 1/D (i = 1 \text{ to } N) \), and \( w_{im}^2 = d_{kp} \)
\( (p = 1 \text{ to } P) \).

adjust output function \( g_{ij}( ) \) \( (i = 1 \text{ to } N, j = 1 \text{ to } m) \) according to the following rule:

IF \( (| \Theta_{im} - \Theta_{ij} | = \text{MAX} | \Theta_{im} - \Theta_{ij} |, | \Theta_{im} - \Theta_{iu} | \neq 0 \), \( 1 \leq i \leq N \)
and \( | \Theta_{im} - \Theta_{ij} | = \text{MIN} | \Theta_{im} - \Theta_{ij} | \), \( 1 \leq j \leq m-1 \), THEN

\[ IF ((\Theta_{im} - \Theta_{ij}) > 0), \quad \text{THEN} \]
set $\alpha_{in}^1 = 1/(\Theta_{in} - \Theta_u)$ if $\alpha_{in}^1 < 1/(\Theta_{in} - \Theta_u)$

set $\alpha_{in}^2 = 1/(\Theta_{in} - \Theta_u)$ if $\alpha_{in}^2 < 1/(\Theta_{in} - \Theta_u)$

ELSE IF $((\Theta_{in} - \Theta_u) < 0)$, THEN

set $\alpha_{in}^2 = 1/(\Theta_u - \Theta_{in})$ if $\alpha_{in}^2 < 1/(\Theta_u - \Theta_{in})$

set $\alpha_{in}^1 = 1/(\Theta_u - \Theta_{in})$ if $\alpha_{in}^1 < 1/(\Theta_u - \Theta_{in})$

END IF

END IF

increment $k$

END DO UNTIL

DO UNTIL (the largest output error $\sigma < E_o$)

initialize $k=1$

DO UNTIL ($k \geq K$)

input the $k/h$ training samples and compute the output error

IF (there is one $|e_p| > E_i$ (p=1 to P)), THEN

add a new MIN-FN in layer 2 and increment $m$

set $\Theta_i = x_i$, $\alpha_{in}^1 = 1/D$, $\alpha_{in}^2 = 1/D$ (i=1 to $N$), and $w_{ip}^2 = d_{ip}$

(p=1 to P).

adjust output function $g_\phi(\ )$ (i=1 to $N$, j=1 to $m$) according to the following rule:

99
\[
\begin{align*}
\text{IF} \ (|\Theta_{lm}-\Theta_i| = \max_{1 \leq i \leq N} |\Theta_{lm}-\Theta_i|, |\Theta_{lm}-\Theta_u| \neq 0) \\
\text{and} \ (|\Theta_{lm}-\Theta_u| = \min_{1 \leq j \leq m-1} |\Theta_{lm}-\Theta_j|), \ \text{THEN} \\
\text{IF} \ ((\Theta_{lm}-\Theta_u) > 0), \ \text{THEN} \\
\text{set} \ \alpha^1_{lm} = 1/(\Theta_{lm}-\Theta_u) \ \text{if} \ \alpha^1_{lm} < 1/(\Theta_{lm}-\Theta_u) \\
\text{set} \ \alpha^2_{lm} = 1/(\Theta_{lm}-\Theta_u) \ \text{if} \ \alpha^2_{lm} < 1/(\Theta_{lm}-\Theta_u) \\
\text{ELSE IF} \ ((\Theta_{lm}-\Theta_u) < 0), \ \text{THEN} \\
\text{set} \ \alpha^2_{lm} = 1/(\Theta_{lm}-\Theta_u) \ \text{if} \ \alpha^2_{lm} < 1/(\Theta_{lm}-\Theta_u) \\
\text{set} \ \alpha^1_{lm} = 1/(\Theta_{lm}-\Theta_u) \ \text{if} \ \alpha^1_{lm} < 1/(\Theta_{lm}-\Theta_u) \\
\end{align*}
\]

END IF

END IF

\[
\sigma = \max_k (|e_k|)
\]
increment \(k\)

END DO UNTIL

\[
\sigma = \max_k (|\sigma_k|)
\]

END DO UNTIL

END
By using these self-organizing learning algorithms, fuzzy inference network classifiers are constructed during the training procedure. The learning process is a procedure of selecting prototypes from training data and determining inference rules by adjusting output functions and weights of the fuzzy neurons. Since the prototypes are selected in such a way that the resultant networks can recall all the training patterns within the error tolerance $E_r$, the learning is always convergent because the extreme case is to select all the training patterns as prototypes. The learning will finish in a few epochs.

The proposed classification algorithms and learning algorithms allow overlapping for the hyperboxes and the fuzzy classes. Fuzzy boundaries of the fuzzy classes can be found by training the networks using fuzzy or nonfuzzy labelled data.
4.4 Analyses of the FIN Classifiers and Learning Algorithms

4.4.1. Analysis of the FIN classifiers

MMFIN and MSFIN have been used as fuzzy classifiers. After being trained, they can estimate the membership values of the input patterns in each fuzzy class and find the fuzzy partitions.

The first layers of the FIN classifiers accept data into the networks. The output functions of each TRAN-FN in the first layers then transfer the input data into membership values. These membership values describe the distances or the similarities of each input to the corresponding feature value of the prototypes.

Because the output functions of the TRAN-FNs are triangle membership functions, the FIN classifiers create a triangle fuzzy number for each feature value of a prototype. Consequently, the FIN classifiers establish a fuzzy pattern in their structures for each prototype. The thresholds and slope parameters, \( t^1 \), \( t^2 \) and \( \alpha \) for MMFIN classifier, or \( \alpha^1 \) and \( \alpha^2 \) for the MSFIN classifier, control the fuzzy extent of the corresponding prototype. The way of selecting thresholds and
slope parameters will affect the computation of membership values and the number of MIN-FNs in the second layer.

The MIN-FNs in the second layers represent prototypes. The minimum operations of the MIN-FNs make fuzzy "AND" on all the fuzzy outputs from the first layers. Each MIN-FN represents a fuzzy hyperbox. The fuzzy hyperboxes are fuzzy sets defined in the hyperboxes in the pattern space $U \subseteq \mathbb{R}^N$. The outputs of the MIN-FNs give the membership values of the input pattern to the fuzzy hyperboxes that describe how much the input pattern is similar to the prototypes. The membership function defined on each hyperbox is:

$$c_i(X) = \text{MIN}[g_{i1}(x_1), g_{i2}(x_2), \ldots, g_{iN}(x_N)]$$

where $g_{ij}(x_j)$ ($i = 1, 2, \ldots, N$) are the output functions of the $ith$ TRAN-FN. So the membership functions of the hyperboxes are determined by the output functions. Fig. 4.6 (a) and (b) give the two membership functions defined on two-dimensional hyperboxes for MMFIN and MSFIN, respectively.
(a) hyperbox membership function for MMFIN

(b) hyperbox membership function for MSFIN

Fig. 4.6 Hyperbox membership functions defined in 2-D hyperboxes
The output functions of the TRAN-FNs for the MMFIN classifier are triangular functions with determined slopes and variable thresholds, and those functions for the MSFIN classifier are triangular functions with variable slopes and zero thresholds. The slope and threshold parameters of the output functions determine the boundaries of hyperboxes.

MAX-FNs are used in the third layer of MMFIN. The weighted inputs of the MAX-FNs contain the information of the input pattern similar to each prototype and the information of each prototype belonging to all the fuzzy classes. The maximum operations of MAX-FNs choose the maximum weighted input as the membership values of the input pattern belonging to each fuzzy class. The Min-Max fuzzy inference network implements a "most important prototype" classification algorithm.

In the MSFIN classifier, SUM-FNs are used in the third layer. Now the summations of the SUM-FNs combine all the information of the input pattern similar to each prototype and the information of the prototypes belonging to all the fuzzy classes. Weighted averages are given as the membership values of the input pattern belonging to all the fuzzy classes. The Min-Sum fuzzy inference network implements an "important prototypes" classification algorithm.
The proposed FIN classifiers have neural network structures and process information using fuzzy algorithms. They can learn from environments, find fuzzy and hard partitions, and classify patterns with overlapping characteristics.

4.4.2. Analysis of the learning algorithms

The proposed learning algorithms are self-organizing algorithms. During the learning procedure, the structures and the parameters of the FIN classifiers are determined. The numbers of TRAN-FNs in the first layers and the numbers of MAX-FNs or SUM-FNs in the third layers are determined, respectively, by the numbers of inputs and outputs of training samples. The numbers of MIN-FNs in the second layers are determined by the learning algorithms, E, and the training patterns.

The proposed self-organizing learning algorithms are improved leader clustering algorithms. The first stage of the learning is leader clustering. The learning algorithms select the first training pattern as the first prototype. The next training pattern is compared with the first prototype. It follows the first prototype if the maximum output error is less than the error tolerance E.
Otherwise, it is selected as a new prototype. This process is repeated for all following training patterns. In the second stage of the learning, each training pattern is fed to the network to check the output error. If the maximum output error is greater than $E_t$, a new prototype is added. The second stage is repeated until the maximum output error is less than or equal to $E_t$ for all the training patterns. The information for prototypes is stored in the structure of the network in the form of fuzzy inference rules. The number of prototypes thus grows with time and depends on $E_t$, the similarities between the training patterns as well as the method of computing outputs. The proposed self-organizing learning algorithms are better than the leader clustering since the second stage is added to ensure that the maximum error for training patterns is less than $E_t$.

The proposed fuzzy inference networks are trained to match all the training patterns within the error tolerance, i.e., the criterion of the proposed learning algorithms is:

$$
\sigma = \max_{k=1}^{K} \max_{p=1}^{P} \left| y_{kp} - d_{kp} \right| \leq E_t \tag{4.12}
$$

where $\sigma$ is the maximum estimated error, $d_{kp}$ is the desired output and $y_{kp}$ is the network output for the $p$th fuzzy class of the $k$th training pattern. The training
will be finished in a few epochs. This is different from those backpropagation type learning algorithms, which adjust parameters to minimize the error \( E = (d(t) - y(t))^2 \) in such a way that:

\[
\Delta w = -\eta \frac{\partial E}{\partial w}
\]  

(4.13)

where \( d(t) \) is the desired output, \( y(t) \) is the current output and \( w \) is the adjustable parameter. In backpropagation type learning algorithms, the training procedure has to be performed for many epochs in order to achieve the optimal results. The training of the proposed FIN classifiers is much faster than the training of those networks using backpropagation learning.

When learning a pattern, the Min-Max fuzzy inference network computes the actual network outputs first, and then the actual outputs are compared with the desired outputs. If one of the actual outputs is much larger than the desired output (say \( y^3_p - d_{pk} > E \)), the threshold parameters of the output functions of the TRAN-FNs have to be adjusted to reduce the actual outputs and to reduce the important hyperboxes. This procedure will be repeated until \( y^3_p - d_{pk} \leq E \). If one of the actual outputs is much less than the desired output (say \( y^3_p - d_{pk} < -E \)), the training pattern is selected as a new prototype and a new MIN-FN is established.
to represent it. If the error is acceptable (say $| y_{p}^{3} - d_{pk} | \leq E$), this training pattern is treated as a pattern that is similar to one of the prototypes and it is not necessary to remember its information. The number of MIN-FNs in the second layer can be determined only after the learning procedure is finished. It depends on $E_i$ and the similarities between training patterns.

The learning process of the Min-Sum fuzzy inference network is: firstly, it computes the actual network outputs and then compares them with the desired outputs. If the maximum output error is much larger than the desired output (say $| y_{p}^{3} - d_{pk} | > E$), the training pattern is selected as a new prototype and a new MIN-FN is established to represent it. Then the slope parameters of the output functions of the TRAN-FNs are adjusted to reduce some of the hyperboxes. If the error is acceptable (say $| y_{p}^{3} - d_{pk} | \leq E$), this training pattern is treated as a pattern that is similar to one of the prototypes. The number of MIN-FNs in the second layer can be determined only after the learning procedure is finished. It depends on $E_i$ and the similarities between training patterns. The values of $E_i$ may affect the structures of the FIN classifiers, the learning speed and the classification results of the MMFIN classifier.
4.5 Simulations

The proposed FIN classifiers were simulated on a PC-486 (33MHz) using C language. In the simulations, the proposed FIN classifiers were trained by four sets of two-dimensional training samples (Training Data Sets 1, 2, 3 and 4).

Training Data Set 1 and Training Data Set 2 (given in Tables 4.1 and 4.2) are training samples with fuzzy labels from the "fuzzy Exclusive OR" (or a quasi-XOR problem) [Enbustsu et al., 1991]. There are 2 fuzzy classes in the quasi-XOR problem, $Y_1$ and $Y_2$. Training Data Set 2 has more input-output samples than Training Data Set 1. Fuzzy labels are available in practical applications. For example, in pattern recognition problems, the "typicalness" of training samples can be considered and used as fuzzy labels of the training samples. We can also assign class membership values to training pattern according to some rules. Keller et al. proposed three methods to assign class memberships to the training samples [Keller et al., 1985].

Training Data Sets 3 and 4 are samples from the circle-in-the-square problem. The labels in these two sets are nonfuzzy. There are two classes in the square: inside the circle and outside the circle. The area inside the circle equals
to the area outside the circle. This is a typical classification problem. There are 100 samples in Training Data Set 3 and 1000 samples in Training Data Set 4. These samples are randomly distributed in the square.

The training samples from the Fuzzy Exclusive OR problem are used to test fuzzy classification ability and the ability to extract fuzzy inference rules from training data of the FIN classifiers. The training samples from the circle-in-the-square problem are used to test the classification ability of the FIN classifiers.

In the simulation experiments, different values of learning parameters were selected to investigate their effects on the FIN classifiers. The simulation results of the MMFIN classifier are listed in Table 4.3. The simulation results of the MSFIN classifier are listed in Table 4.4. In Tables 4.3 and 4.4, N, M, and P represent, respectively, the numbers of FNs in the first, the second, and the third layers. t is the learning time (in seconds). σ is the maximum estimated error. R is the number of epochs of the training samples being fed to the networks.

Fuzzy inference rules are established during the learning procedure. The membership functions of the inference rules are automatically determined during the learning. Table 4.5 gives the fuzzy inference rules extracted by MMFIN.
when trained by Training Data Set 1. Table 4.6 gives the fuzzy inference rules extracted by MSFIN when trained by Training Data Set 1. The membership functions of the inference rules are given in Fig. 4.7.

For illustration, the classification results of the FIN classifiers are compared with those of the fuzzy ARTMAP [Carpenter et al., 1992]. Table 4.7 gives the comparison results of the MMFIN and the MSFIN with the fuzzy ARTMAP.
### TABLE 4.1 Training Data Set 1

9 training samples of fuzzy exclusive OR problem

<table>
<thead>
<tr>
<th>$x_2$</th>
<th>$x_1$</th>
<th>$0.0$ (Y^1, Y^2)</th>
<th>$0.5$ (Y^1, Y^2)</th>
<th>$1.0$ (Y^1, Y^2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0, 1.00</td>
<td>0, 0</td>
<td>1.00, 0</td>
</tr>
<tr>
<td>0.5</td>
<td>0.0</td>
<td>0, 0</td>
<td>0, 0</td>
<td>0, 0</td>
</tr>
<tr>
<td>1.0</td>
<td>1.00, 0</td>
<td>0, 0</td>
<td>0, 0</td>
<td>0, 1.00</td>
</tr>
</tbody>
</table>

### TABLE 4.2 Training Data Set 2

36 training samples with fuzzy labels of fuzzy exclusive OR problem

<table>
<thead>
<tr>
<th>$x_2$</th>
<th>$x_1$</th>
<th>$0.0$ (Y^1, Y^2)</th>
<th>$0.2$ (Y^1, Y^2)</th>
<th>$0.4$ (Y^1, Y^2)</th>
<th>$0.6$ (Y^1, Y^2)</th>
<th>$0.8$ (Y^1, Y^2)</th>
<th>$1.0$ (Y^1, Y^2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0, 1.00</td>
<td>0, 0.75</td>
<td>0, 0.25</td>
<td>0.25, 0</td>
<td>0.75, 0</td>
<td>1.00, 0</td>
</tr>
<tr>
<td>0.2</td>
<td>0.0</td>
<td>0, 0.75</td>
<td>0, 0.75</td>
<td>0, 0.25</td>
<td>0.25, 0</td>
<td>0.75, 0</td>
<td>0.75, 0</td>
</tr>
<tr>
<td>0.4</td>
<td>0.0</td>
<td>0, 0.25</td>
<td>0, 0.25</td>
<td>0, 0.25</td>
<td>0.25, 0</td>
<td>0.25, 0</td>
<td>0.25, 0</td>
</tr>
<tr>
<td>0.6</td>
<td>0.25, 0</td>
<td>0.25, 0</td>
<td>0.25, 0</td>
<td>0.25, 0</td>
<td>0.25, 0</td>
<td>0, 0.25</td>
<td>0, 0.25</td>
</tr>
<tr>
<td>0.8</td>
<td>0.75, 0</td>
<td>0.75, 0</td>
<td>0.25, 0</td>
<td>0.25, 0</td>
<td>0.25, 0</td>
<td>0.75, 0</td>
<td>0, 0.75</td>
</tr>
<tr>
<td>1.0</td>
<td>1.00, 0</td>
<td>0.75, 0</td>
<td>0.25, 0</td>
<td>0.25, 0</td>
<td>0.25, 0</td>
<td>0.75, 0</td>
<td>0, 1.00</td>
</tr>
</tbody>
</table>

113
### TABLE 4.3 Learning results of the MMFIN classifier

<table>
<thead>
<tr>
<th>Training Set</th>
<th>( E_i )</th>
<th>N</th>
<th>M</th>
<th>P</th>
<th>t(sec)</th>
<th>( \sigma )</th>
<th>R</th>
<th>error rate</th>
<th>Fig. No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training Data Set 1</td>
<td>0.1</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>0.05</td>
<td>0.00</td>
<td>2</td>
<td></td>
<td>Fig.4.8</td>
</tr>
<tr>
<td>Training Data Set 2</td>
<td>0.1</td>
<td>2</td>
<td>14</td>
<td>2</td>
<td>0.07</td>
<td>0.10</td>
<td>2</td>
<td></td>
<td>Fig.4.9</td>
</tr>
<tr>
<td>Training Data Set 3</td>
<td>0.1</td>
<td>2</td>
<td>40</td>
<td>2</td>
<td>1.43</td>
<td>0.10</td>
<td>3</td>
<td>8.60%</td>
<td>Fig.4.12a</td>
</tr>
<tr>
<td>Training Data Set 3</td>
<td>0.2</td>
<td>2</td>
<td>20</td>
<td>2</td>
<td>1.79</td>
<td>0.20</td>
<td>4</td>
<td>12.1%</td>
<td>Fig.4.12b</td>
</tr>
<tr>
<td>Training Data Set 4</td>
<td>0.1</td>
<td>2</td>
<td>91</td>
<td>2</td>
<td>7.31</td>
<td>0.10</td>
<td>4</td>
<td>3.37%</td>
<td>Fig.4.13a</td>
</tr>
<tr>
<td>Training Data Set 4</td>
<td>0.2</td>
<td>2</td>
<td>62</td>
<td>2</td>
<td>4.12</td>
<td>0.20</td>
<td>5</td>
<td>4.53%</td>
<td>Fig.4.13b</td>
</tr>
</tbody>
</table>

### TABLE 4.4 Learning results of the MSFIN classifier

<table>
<thead>
<tr>
<th>Training Set</th>
<th>( E_i )</th>
<th>N</th>
<th>M</th>
<th>P</th>
<th>t(sec)</th>
<th>( \sigma )</th>
<th>R</th>
<th>error rate</th>
<th>Fig. No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training Data Set 1</td>
<td>0.1</td>
<td>2</td>
<td>8</td>
<td>2</td>
<td>0.05</td>
<td>0.00</td>
<td>2</td>
<td></td>
<td>Fig.4.10</td>
</tr>
<tr>
<td>Training Data Set 2</td>
<td>0.1</td>
<td>2</td>
<td>31</td>
<td>2</td>
<td>0.22</td>
<td>0.00</td>
<td>3</td>
<td></td>
<td>Fig.4.11</td>
</tr>
<tr>
<td>Training Data Set 3</td>
<td>0.1</td>
<td>2</td>
<td>42</td>
<td>2</td>
<td>1.32</td>
<td>0.09</td>
<td>3</td>
<td>5.98%</td>
<td>Fig.4.14a</td>
</tr>
<tr>
<td>Training Data Set 3</td>
<td>0.2</td>
<td>2</td>
<td>33</td>
<td>2</td>
<td>1.26</td>
<td>0.18</td>
<td>3</td>
<td>6.72%</td>
<td>Fig.4.14b</td>
</tr>
<tr>
<td>Training Data Set 4</td>
<td>0.1</td>
<td>2</td>
<td>197</td>
<td>2</td>
<td>19.4</td>
<td>0.00</td>
<td>3</td>
<td>2.22%</td>
<td>Fig.4.15a</td>
</tr>
<tr>
<td>Training Data Set 4</td>
<td>0.2</td>
<td>2</td>
<td>174</td>
<td>2</td>
<td>24.6</td>
<td>0.20</td>
<td>4</td>
<td>2.23%</td>
<td>Fig.4.15b</td>
</tr>
</tbody>
</table>

114
### TABLE 4.5 Fuzzy inference rules extracted by MMFIN when trained by Training Data Set 1

<table>
<thead>
<tr>
<th>Rule</th>
<th>IF:</th>
<th>THEN:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$x_1$ is small and $x_2$ is small</td>
<td>$y_1$ is 0 and $y_2$ is 1</td>
</tr>
<tr>
<td>2</td>
<td>$x_1$ is small and $x_2$ is large</td>
<td>$y_1$ is 1 and $y_2$ is 0</td>
</tr>
<tr>
<td>3</td>
<td>$x_1$ is large and $x_2$ is small</td>
<td>$y_1$ is 1 and $y_2$ is 0</td>
</tr>
<tr>
<td>4</td>
<td>$x_1$ is large and $x_2$ is large</td>
<td>$y_1$ is 0 and $y_2$ is 1</td>
</tr>
</tbody>
</table>

### TABLE 4.6 Fuzzy inference rules extracted by MSFIN when trained by Training Data Set 1

<table>
<thead>
<tr>
<th>Rule</th>
<th>IF:</th>
<th>THEN:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$x_1$ is small and $x_2$ is small</td>
<td>$y_1$ is 0 and $y_2$ is 1</td>
</tr>
<tr>
<td>2</td>
<td>$x_1$ is small and $x_2$ is medium</td>
<td>$y_1$ is 0 and $y_2$ is 0</td>
</tr>
<tr>
<td>3</td>
<td>$x_1$ is small and $x_2$ is large</td>
<td>$y_1$ is 1 and $y_2$ is 0</td>
</tr>
<tr>
<td>4</td>
<td>$x_1$ is medium and $x_2$ is small</td>
<td>$y_1$ is 0 and $y_2$ is 0</td>
</tr>
<tr>
<td>5</td>
<td>$x_1$ is medium and $x_2$ is large</td>
<td>$y_1$ is 0 and $y_2$ is 0</td>
</tr>
<tr>
<td>6</td>
<td>$x_1$ is large and $x_2$ is small</td>
<td>$y_1$ is 1 and $y_2$ is 0</td>
</tr>
<tr>
<td>7</td>
<td>$x_1$ is large and $x_2$ is medium</td>
<td>$y_1$ is 0 and $y_2$ is 0</td>
</tr>
<tr>
<td>8</td>
<td>$x_1$ is large and $x_2$ is large</td>
<td>$y_1$ is 0 and $y_2$ is 1</td>
</tr>
</tbody>
</table>
Fig. 4.7(a) Membership functions of the extracted inference rules by MMFIN
Fig. 4.7(b) Membership functions of the extracted inference rules by MSFIN
TABLE 4.7 Classification results of FIN classifiers compared with Fuzzy ARTMAP for Circle-in-the-Square problem

<table>
<thead>
<tr>
<th></th>
<th>MMFIN</th>
<th>MSFIN</th>
<th>Fuzzy ARTMAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Training exemplars</td>
<td>100</td>
<td>1000</td>
<td>100</td>
</tr>
<tr>
<td>Training epochs</td>
<td>3</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Recalling Rate on Testing Set</td>
<td>91.4%</td>
<td>96.6%</td>
<td>94.0%</td>
</tr>
</tbody>
</table>

Fig. 4.8 and Fig. 4.9 show the fuzzy partitions for the Fuzzy Exclusive OR problem by the MMFIN classifier. Fig. 4.10 and Fig. 4.11 show the fuzzy partitions for the Fuzzy Exclusive OR problem by the MSFIN classifier.

Fig. 4.12 and Fig. 4.13 are the classification results of the MMFIN classifier for the circle-in-the-square problem with different learning parameters and trained by different numbers of training samples. Fig. 4.14 and Fig. 4.15 are the classification results of the MSFIN classifier for the circle-in-the-square problem. Fig. 4.16 gives the classification results of fuzzy ARTMAP when trained by different number of training samples.
a. Fuzzy boundary of fuzzy class $Y^1$

b. Fuzzy boundary of fuzzy class $Y^2$

Fig. 4.8 Fuzzy partitions for the Fuzzy Exclusive OR problem by the MMFIN classifier (9 training samples) problem
a. Fuzzy boundary of fuzzy class $Y^1$

b. Fuzzy boundary of fuzzy class $Y^2$

Fig. 4.9 Fuzzy partitions for the Fuzzy Exclusive OR problem by the MMFIN classifier (36 training samples)
Fig. 4.10 Fuzzy partitions for the Fuzzy Exclusive OR problem by the MSFIN classifier (9 training samples) problem
Fig. 4.11 Fuzzy partitions for the Fuzzy Exclusive OR problem by the MSFIN classifier (36 training samples)
Fig. 4.12 Classification results of the MMFIN classifier for the circle-in-the-square problem (100 training samples)
Fig. 4.13 Classification results of the MMFIN classifier for the circle-in-the-square problem (1000 training samples)
Fig. 4.14 Classification results of the MSFIN classifier for the circle-in-the-square problem (100 training samples)
Fig. 4.15 Classification results of the MSFIN classifier
for the circle-in-the-square problem (1000 training samples)
Fig. 4.16 Classification results of fuzzy ARTMAP for the circle-in-the-square problem
From the simulation results, we note that the smaller $E_i$ is, the more MIN-FNs are needed in the second layer and the classification results are better. However, the number of the MIN-FNs will not exceed the number of training patterns. The value of $E_i$ affects the number of MIN-FNs in the second layer, and the classification results. If more training samples are learned, the classification results are better. MMFIN needs fewer number of MIN-FNs in the second layer than MSFIN. However, MSFIN can get smoother membership functions for fuzzy classification problems and lower error rate for nonfuzzy classification problems than MMFIN when trained by the same set of training data. The learning of the proposed FIN classifiers is very fast and can be finished in a few epochs.

From the recognition rates of the FIN classifiers and those of the fuzzy ARTMAP, we note that the classification results of the FIN classifiers are better than those of the fuzzy ARTMAP.
4.6 Conclusions

Two FIN classifiers have been used as classifiers. Efficient self-organizing learning algorithms for the FIN classifiers have also been presented. The proposed FIN classifiers can learn fuzzy inference rules and their membership functions and then determine fuzzy and hard partitions according to the membership values. The proposed FIN classifiers have the following advantages:

1. The ability to represent the knowledge of the network by a set of fuzzy inference rules.
2. Automatic acquisition of fuzzy or nonfuzzy partitions from training data.
3. Dynamic adaptation to environments through self-organizing the structures.
5. Better classification results than fuzzy ARTMAP.

The proposed FIN classifiers can be used in applications where an intelligent decision has to be made on the classes of a given pattern. Potential
applications of the FIN classifiers include fuzzy decision systems such as pattern recognition systems or control systems.
CHAPTER V

FUZZY INFERENCE NETWORKS FOR PATTERN RECOGNITION

Fig. 5.1 shows a general scheme of pattern recognition system. There are two important steps in pattern recognition: feature extraction and classification. In Chapter 4, MMFIN and MSFIN were used as classifiers. They can be used in pattern recognition systems as the classification parts. In this case, important features of the patterns have to be extracted by appropriate feature extraction methods before the classification.

In this chapter, improved Min-Competitive Inference Networks (MCFIN-A and MCFIN-B) are developed as letter recognition systems. In contrast to most of the conventional pattern recognition systems, they do not require any preprocessing of feature extraction. Instead, the feature extraction step is incorporated in the structure of the networks. The proposed MCFINs can recognize image patterns without any preprocessing.
Fig. 5.1 A general scheme of pattern recognition system
5.1 Pattern Recognition Systems Insensitive to Shifts and Distortions of Input Patterns

Most of the methods of pattern recognition, for example, pattern matching, are oversensitive to shifts in position and deformations in shape of the stimulus patterns. It is necessary to normalize the position and the shape of the stimulus patterns. A good method for normalization, however, has not yet been developed. Therefore, it is desirable to find algorithms that can cope with the shifts in position and distortions in shape.

Due to their generalization ability, neural networks have been used in pattern recognition problems, especially when the input patterns are shifted in position, scale-changed and distorted. Fukushima et al. [Fukushima et al., 1989, 1991] presented the Neocognitron, which is insensitive to translation and deformation of input patterns and was used to recognize hand-printed characters. However, the Neocognitron is complex and needs many cells. Many layers are used for feature extraction and deformation toleration. The training is very complicated. Carpenter and Grossberg [Carpenter and Grossberg, 1988] proposed a self-organizing system that can classify patterns by adaptive resonance
theory. However, a lot of internal exemplars including noise patterns are formed in the network. Guyon et al. [Guyon et al., 1991] designed a system for on-line recognition of handwritten characters for a touch terminal using time-delay neural networks and the BP algorithm. Time information is used in their approach due to the special application environment. Preprocessings including resampling, centering position and rescaling are performed before training and recognition. This system needs many training patterns for learning before it can work properly. Fukumi et al. [Fukumi et al., 1992] proposed a neural pattern recognition system trained by the BP algorithm that can be used to recognize rotated patterns. Normalization was performed to find the centre of the coin and rotated coin images in 72 directions were used to train the network. The major problem of the BP algorithm is that it needs a long training time. Perantonis and Lisboa [Perantonis and Lisboa, 1992] constructed a pattern recognition system that is invariant to translation, rotation and the scale of the input pattern by high-order neural networks. However, the number of weights increases greatly with the order of the network which increases the complexity of the network. Although these invariant neural network pattern recognition systems achieve some success, there is not yet a useful system that is insensitive to shifts in position, distortions in shape, scale-changes and rotation. The problems of the present neural network pattern recognition systems are the long training time and complex
network structures.

Other approaches for invariant pattern recognition use the similarity criterion defined in terms of certain mathematical functions, such as Fourier transform, moment invariance, or similarity to fuzzy set. Fuzzy set theory has proved itself to be of significant importance in pattern recognition problems [Bezdek, 1981; Bezdek and Pal, 1992; Kandel, 1982, Pal and Majumder, 1986]. Fuzzy methods are particularly useful when it is not reasonable to assume class density functions and statistical independence of features. Siy and Chen applied fuzzy logic to handwritten numerical character recognition [Siy and Chen, 1974]. Preprocessing of abstracting position, scale and orientation features of the patterns was used in the approach. Wang and Mendel developed a fuzzy algorithm for hand-written rotation-invariant character recognition [Wang and Mendel, 1992A]. Fuzzy sets for the training patterns and their rotated versions were generated. Input patterns were classified based on the memberships of the patterns to the fuzzy sets.

In this chapter, Min-Competitive Fuzzy Inference Networks (MCFIN-A and MCFIN-B) are developed for letter recognition problems. MCFIN-A is a network that can be used for supervised pattern recognition problems and
MCFIN-B is a network that can be used for unsupervised pattern recognition problems. Corresponding self-organizing learning algorithms for MCFIN-A and MCFIN-B are also presented. The improved Min-Competitive fuzzy inference networks have two fuzzification layers. In contrast to most of the conventional pattern recognition systems, they do not require any preprocessing such as normalizing the position, size, and deformation. Only the typical exemplar patterns are needed for learning. The 26 English letters and the 10 Arabic numerals of standard fonts, each represented by 16X16 pixels, were used as training patterns. Shifted and distorted patterns of these 36 exemplars, and three other sets of alphabets and numerals, including bold fonts, italic fonts and non-standard fonts, were also used to test the performances of the Min-Competitive fuzzy inference networks. The MCFINs can recall all the training patterns. They can also recognize the shifted patterns, distorted patterns and the three other fonts of the standard patterns with high recognition rates.
5.2. Min-Competitive Fuzzy Inference Classification Algorithm

In Chapter 4, Min-Max inference classification and Min-Sum inference classification algorithms were developed and tested by two-dimensional data. In this section, the Min-Competitive inference classification algorithm based on the Min-Competitive inference model will be discussed. The fuzzy inference rules are the same as in Chapters 3.

Min-Competitive Fuzzy Inference Classification Algorithm

BEGIN

input pattern \( X = \{x_1, x_2, ..., x_N\} \), of unknown classification

initialize \( j = 1 \)

DO UNTIL ( \( j > M \))

the membership values of \( X \) to the fuzzy hyperboxes are computed as

\[
c_j = \text{MIN} \{ m_{ij}(x_j) \}_{i=1}^N
\]

137
where \( m_{ij}(\cdot) \) is the membership function of fuzzy set \( A_{ij} \).

increment \( j \)

**END DO UNTIL**

initialize \( p = 1 \)

**DO UNTIL** ( \( p > P \))

the classification features of \( X \) follow the most similar prototype:

\[
y_p = y_{jp}, \quad 1 \leq J \leq M, \quad \text{if} \quad c_j = \underset{j=1}{\overset{M}{\text{MAX}}} c_j
\]

where \( y_{jp} \) is the membership value of the prototype \( X_i \) to class \( F_p \).

increment \( p \)

**END DO UNTIL**

**END**

Similar to Min-Max and Min-Sum inference classifications, the IF part of an inference rule represents a fuzzy hyperbox when \( m_{ij}(\cdot) \) (\( i = 1 \) to \( N \), \( j = 1 \) to \( M \)) are triangular functions. The classification of the input pattern follows the most similar prototype. The "most similar prototype" for an input pattern is defined as \( c_j = \max_{1 \leq j \leq M} (c_j) \). The Min-Competitive fuzzy inference classification
algorithm is a fuzzy version of one-nearest prototype classification [Bezdek et al., 1977]. The nearest prototype classification is widely used in many applications. Many algorithms based on it have been developed and have proved to be useful. However, Min-Competitive inference networks that will be presented in this Chapter are different from the previous nearest prototype classification algorithms in which all the input patterns are fuzzified before the training and recognition procedures. Two fuzzification layers are used to extract important features of the patterns. The fuzzification allows the networks to focus on the important features of the patterns so that the networks have the ability to recognize noisy patterns.

In addition to fuzzification, the proposed "most similar prototype" classification method has another advantage. Because it gives the similarities of the input pattern to each class, statistical data of the prototypes can be also used to help in making decisions. This can be done by ranking the membership values of all the classes, if the top ones are too close so that a decision cannot be made, (for example, there are two maximum outputs, or the differences between the top membership values are less than $E/2$), then statistical data of the prototypes can be used to select the right class among those with top membership values. In this case, the expected recognition results are better than those in which only distance or similarity is used to classify patterns.
Since the Min-Competitive inference method classifies patterns by finding the most similar prototype, it is possible to use it as an unsupervised algorithm by selecting prototypes from input patterns and establishing a cluster for each prototype. Unsupervised learning becomes important when supervised learning is not available.
5.3. Supervised Min-Competitive Fuzzy Inference Network (MCFIN-A)

5.3.1. The structure of MCFIN-A

The MCFIN-A is a four-layer feedforward network (as shown in Fig. 5.2) which is developed for 2-D visual pattern recognition problems. The first layer is the input layer that accepts patterns into the network. TRAN-FNs are used in this layer. Each TRAN-FN in this layer corresponds to one pixel of an input pattern. The FNs are displayed and indexed in two-dimensions and the number of FNs in this layer is equal to the total number of pixels of an input pattern. Assuming each input pattern has $N_1 \times N_2$ pixels, then the first layer has $N_1 \times N_2$ TRAN-FNs. The algorithm of the TRAN-FNs in the first layer is:

\[ y_{ij}^1 = s_{ij}^1 = x_{ij} \quad i = 1, 2, \ldots, N_1, \ j = 1, 2, \ldots, N_2 \]  \hspace{1cm} (5.1)

where $x_{ij}$ is the $(i,j)$th pixel value of an input pattern.
Fig. 5.2 Supervised Min-Competitive Fuzzy Inference Network (MCFIN-A)
The second layer is also displayed in two-dimensions and consists of \( N_1 \times N_1 \) MAX-FNs. This layer fuzzifies input patterns. The state of the \((p,q)th\) MAX-FN in the second layer is:

\[
s_{pq}^2 = \max_{i=1}^{N_1} \max_{j=1}^{N_2} (w^1(i-p,j-q) \cdot y_{ij}^1)
\]

\[
p = 1, 2, \ldots, N_1, \quad q = 1, 2, \ldots, N_2
\]

\[
w^1(m,n) = \exp(-\beta^2(m^2 - n^2))
\]

\[
y_{pqm}^2 = s_{pqm}(s_{pq}^2) = \begin{cases} 
1 - \alpha |s_{pq}^2 - \theta_{pqm}| & \text{if } 1 \geq \alpha |s_{pq} - \theta_{pqm}| > 0 \\
0 & \text{otherwise}
\end{cases}
\]

\[
p = 1, 2, \ldots, N_1; \quad q = 1, 2, \ldots, N_2; \quad m = 1, 2, \ldots, M
\]

where \(w^1(i-p,j-q)\) is the weight function from the \((i,j)th\) TRAN-FN in the first layer to the \((p,q)th\) MAX-FN in the second layer and is defined by equation 5.3. A plot of \(w^1(m,n)\) for \(\beta = 0.3\) is shown in Fig. 5.3. By using this weight function, each MAX-FN in the second layer is just like a lens so that it focuses on one pixel of an input pattern but also can "see" the surrounding pixels. How many pixels an MAX-FN can actually "see" is determined by the value of \(\beta\), so
$w^i(m,n)$ is also called the fuzzification function. $y^i_{m}^2$ is the $m$th output of the $(p,q)th$ MAX-FN that is to be connected to the $m$th MIN-FN in the third layer. Each MAX-FN in the second layer has $M$ outputs ($M$ is the number of MIN-FNs in the third layer). The output functions $g_{pqn}(s_{pqn}^2)$ (for every set of $p$, $q$, and $m$) are the membership functions in the IF parts of the inference rules and are to be determined by the learning algorithm. For simplicity, isosceles triangles with heights equal to 1 and slope values equal to $\alpha$ have been chosen as the output functions (as shown in Fig. 5.4). The first two layers are fuzzification layers.

MIN-FNs are used in the third layer. Each MIN-FN in the third layer represents one inference rule, which contains the information of a prototype. Therefore, the number of MIN-FNs in the third layer, $M$, can be determined only after the learning procedure is finished. The algorithm of the MIN-FNs in the third layer is:

\[ y^3_m = s^3_m = \min_{p=1}^{N_1} \min_{q=1}^{N_2} MIN(MIN(y^2_{pqm})) \quad m=1,2,\ldots,M \]  

(5.5)
Fig. 5.3 The fuzzification function

Fig. 5.4 The output function of MAX-FNs in MCFIN-A and MCFIN-B
The fourth layer is the output layer. COMP-FNs are used in this layer, one for each of the P classes. If an input pattern is most similar to the m\textit{th} prototype, then the classification of the input pattern follows the classification of the m\textit{th} prototype. The algorithm of the COMP-FNs in the fourth layer is:

\[ y_n^4 = s_n^4 = w_j^3 \text{ if } y_j^3 = t^4, \quad n = 1, 2, ..., P \]  
\[ t^4 = \frac{\sum_{m=1}^{M} y_m^3}{M} \]  

where \( w_j^3 \) is the weight from the J\textit{th} MIN-FN in the third layer to the \( n \textit{th} \) COMP-FN in the fourth layer. It contains the information of the J\textit{th} prototype belonging to the \( n \textit{th} \) class and it will be determined in the learning procedure. \( t^4 \) is the activation threshold of all the COMP-FNs in the fourth layer.

5.3.2. Learning algorithm of MCFIN-A

The following parameters have to be determined during the learning procedure: the parameter of the fuzzification function, \( \beta \); the parameters of the output functions of the MAX-FNs in the second layer, \( \alpha \) and \( \Theta_{ijn} \) (i = 1 to \( N_1 \),

146
j = 1 to N_2, and m = 1 to M); the number of MIN-FNs in the third layer, M; and the weights from the FNs in the third layer to the FNs in the fourth layer, w^3_{mn} (m = 1 to M, n = 1 to P). Assuming that there are in total L training patterns in a training set, \( X_k = \{x_{jk}\} \) is the \( k \)th training pattern and \( d_{kn} \) is the \( n \)th desired output of \( X_k \).

**Supervised Learning Algorithm for MCFIN-A:**

**BEGIN**

set \( E_i \), initialize \( \alpha \) and \( \beta \), establish layers 1, 2 and 4

**DO UNTIL** (the largest output error \( \sigma \leq E_i \))

establish the 1st MIN-FN in layer 3 for the 1st training sample

adjust the output functions \( g_{ji}() \) and weights \( w^3_{in} \) by setting \( \Theta_{ji} = x_{ji} \) (i = 1 to N_1, j = 1 to N_2), and \( w^3_{in} = d_{in} \) (n = 1 to P)

initialize m = 1, k = 2

**DO UNTIL** (k \( \geq \) L)

input the \( k \)th training pattern, compute the output errors:

\[ e_n = y^a_n - d_{kn}, \quad (n = 1, 2, ..., P) \]

**IF** (there is one \( |e_n| > E_i \) (n = 1 to P)), **THEN**

add a new MIN-FN in layer 3 and increment m
adjust the output functions $g_{ijm}(\cdot)$ and weights $w^i_{mn}$ by

setting $\Theta_{ijm} = x_{ijk}$ (i = 1 to $N_1$, j = 1 to $N_2$), and $w^i_{mn} = d_{kn}$

($n = 1 \text{ to } P$)

END IF

increment k

END DO UNTIL

initialize k = 1

DO UNTIL (k $\geq L$)

input the $k$th training samples and compute the maximum output error: $\sigma_k = \max_{1 \leq n \leq P} |e_n|

increment k

END DO UNTIL

$\sigma = \max_{1 \leq k \leq L} |\sigma_k|

IF ($\sigma > E_\sigma$) THEN

decrease $\alpha$ and $\beta$ by setting

$\alpha = 0.9\alpha$, $\beta = 0.9\beta$

END IF

END DO UNTIL

END
5.4. Unsupervised Min-Competitive Fuzzy Inference Network (MCFIN-B)

In the previous discussion, all the fuzzy inference networks are supervised learning networks. Fuzzy inference rules are established during the learning procedure. The fuzzy inference networks can then classify patterns according to the learned inference rules. However, supervised learning is not available in some pattern recognition problems. In this section, the structure and learning algorithm of the unsupervised Min-Competitive Fuzzy Inference Network (MCFIN-B) will be developed.

5.4.1. The structure of MCFIN-B

The proposed unsupervised learning fuzzy Min-Competitive fuzzy inference network (MCFIN-B) is shown in Fig. 5.5. The first three layers of the MCFIN-B are the same as the MCFIN-A. TRAN-FNs are used in the input layer to accept patterns and MAX-FNs are used in the second layer to fuzzify patterns. The fuzzification functions and the output functions of the MAX-FNs are also the same as in the MCFIN-A.
Fig. 5.5 Unsupervised Min-Competitive Fuzzy Inference Network (MCFIN-B)
MIN-FNs are used in the third layer of MCFIN-B and the algorithm of the MIN-FNs in the third layer is the same as in MCFIN-A. Now each MIN-FN in the third layer of MCFIN-B represents the prototype of a cluster. It contains the information of a pattern selected as the prototype of a subset consisting of similar training patterns. The number of MIN-FNs in the third layer, \( M \), is determined by the learning procedure.

The fourth layer is the output layer. COMP-FNs are used in this layer of MCFIN-B, one for each of the \( M \) clusters, to provide non-fuzzy outputs. If an input pattern is most similar to the prototype of the \( m \)-th cluster, then the output of the \( m \)-th COMP-FN in the fourth layer is 1 while the outputs of other COMP-FNs are 0. The algorithm of the COMP-FNs in the fourth layer is:

\[
z_m^4 = y_m^3
\]  
(5.8)

\[
s_m^4 = y_m^4 = \begin{cases} 
1 & \text{if } y_m^3 = t^4 \\
0 & \text{if } y_m^3 < t^4 
\end{cases}
\]
(5.9)

\[
m = 1, 2, ..., M
\]

\[
t^4 = \max_{m=1}^M (y_m^3)
\]
(5.10)
5.4.2. Self-organizing learning algorithm

Unsupervised Learning Algorithm for MCFIN-B:

BEGIN

set $E_t$, initialize $\alpha$ and $\beta$, establish layers 1 and 2

establish the 1st MIN-FN in layer 3 for the 1st training sample

adjust the output functions $g_{ij}(\cdot)$ and weights $w_{in}^3$ by setting $\Theta_{ij} = x_{ij}$,

$(i = 1$ to $N_1$, $j = 1$ to $N_2)$, and $w_{in}^3 = d_{in}$ $(n = 1$ to $P)$.

initialize $m = 1$, $k = 2$

DO UNTIL ($k \geq L$)

input the $k$th training pattern, compute the output errors:

$$e_n = y_n^d - d_{nk}, \quad (n = 1, 2, ..., P)$$

IF (there is one $|e_n| > E_t$ $(n = 1$ to $P)$), THEN

add a new MIN-FN in layer 3 and increment $m$

adjust the output functions $g_{ijn}(\cdot)$ and weights $w_{mn}^3$ by setting

$\Theta_{ijn} = x_{ijk}$ $(i = 1$ to $N_1$, $j = 1$ to $N_2)$, and $w_{mn}^3 = d_{kn}$ (for $n = 1$ to $P$)

END IF

increment $k$

END DO UNTIL

END
5.5 Analyses of the Min-Competitive Fuzzy Inference Networks

Two four-layer Min-Competitive Fuzzy Inference Networks have been developed for pattern recognition problems. As other neural networks, they are parallel systems and the network structures are adaptively constructed during the learning procedure.

The first layers of the MCFINs accept input patterns into the networks. The weight functions from the first layer to the second layer fuzzify the position information of the input patterns. Each MAX-FN in the second layer is connected to all the TRAN-FNs in the first layer by the weight function \( w^i(i-p, j-q) \) and takes the maximum value of all the weighted inputs as its state. The result of using MAX-FNs and using fuzzification weights is that one dark pixel in the input pattern will affect the states of several MAX-FNs in the second layer. The second layers of the networks also fuzzify the selected prototypes by using triangular output functions to represent the fuzzy sets in the IF parts of the inference rules. Consequently, the position information of dark pixels and the grey values of all the pixels of the input pattern are fuzzified by the first two layers and this will result in a network that is insensitive to local shifts and distortions of the input patterns. The degree of an input pattern being fuzzified
depends on the parameters $\alpha$ and $\beta$. The smaller the value of $\alpha$ or $\beta$ is, the more FNs in the second layer are affected by a dark pixel of an input pattern. This means that $\alpha$ and $\beta$ control the extent of fuzzification. If $\alpha$ or $\beta$ is too small, the networks cannot separate some distinct training patterns. If $\alpha$ or $\beta$ is too large, the networks may lose the ability to recognize some noisy patterns. $\alpha$ and $\beta$ should be so chosen that all the distinct training patterns can be separated and the networks have good recognition rates.

The $m$th outputs of the $(p,q)th$ MAX-FN in the second layer, $y_{pqm}^2$, expresses the fuzzy concept that "the pixel values around the $(p,q)th$ pixel of the input pattern are similar to the pixel values around the $(p,q)th$ pixel of the $mth$ prototype". The output function $g_{pqm}(\cdot)$ is a membership function. It contains the information of the pixel values around the $(p,q)th$ pixel of the $mth$ prototype.

The first and the second layers perform the feature extraction of the MCFIN pattern recognition systems. The positions of dark pixels are extracted and represented to the next stage of the system, which will classify the input pattern based on the Min-Competitive classification algorithm. In contrast to most pattern recognition systems, the feature extraction is incorporated in the structures of the networks in the proposed MCFINs.
The MIN-FNs in the third layer compute the similarities of the input pattern to the prototypes. The slope parameter $\alpha$ affects the computation of similarities. If the input pattern $X$ is one of the prototypes, one output of the third layer (say $y^3_{m,n}$) will be equal to 1. If the input pattern is not any of the prototypes, all the outputs of the third layer will be less than 1.

The output layer of MCFIN-A gives competitive outputs that indicate which class the input pattern belongs to. The maximum output of the third layer is the activation threshold of all the COMP-FNs in the fourth layer. If $y^3_i$ is the maximum among all the outputs of the third layer, then $y^4_n$ will follow $w^4_{i,n}$.

The output layer of MCFIN-B gives nonfuzzy outputs. It chooses the maximum similarity as the activation threshold of all the COMP-FNs in the fourth layer. If $y^3_m$ is the maximum among all the outputs of the third layer, then the output of the $m$th COMP-FN in the fourth layer is 1 and the outputs of the other COMP-FNs in the fourth layer are 0.

The differences between MCFIN-A and MCFIN-B are the output layers and the learning algorithms. MCFIN-A is a supervised learning network and the third layer of the MCFIN-A is constructed during the learning. The number of
COMP-FNs in the output layer is determined by the number of classes. MCFIN-B is an unsupervised learning network. The third and the fourth layers are constructed during the learning. The number of FNs in these two layers can be determined only after the learning procedure is finished.

The learning of the MCFIN-A is a process of determining prototypes and inference rules from training data. During the learning procedure, if a learned prototype or a pattern similar to one of the prototypes is fed to MCFIN-A, it will be treated as a previous learned pattern without relearning it. If a pattern that is different from the prototypes is fed, it will be treated as a new prototype. A new MIN-FN in the third layer will be established to represent it. Whether a training pattern will be treated as a prototype or as a learned pattern is determined by the similarities of this pattern to all the existing prototypes and by the learning parameters $\alpha$, $\beta$ and $E_i$. If the maximum output error is less than $E_i$, then this input pattern is treated as a previously learned pattern. Otherwise, it is treated as a new prototype.

The learning process of MCFIN-B is a process of determining the clusters and the prototype for each cluster. If a pattern that is similar to one of the prototypes is fed to MCFIN-B (i.e., one of the outputs of the third layer is larger
than or equal to 1-E), it will be treated as a previous learned pattern without relearning it. If a distinct pattern is fed (i.e., all the outputs of the third layer are less than 1-E), it will be treated as the prototype of a new cluster. A new MIN-FN in the third layer and a new COMP-FN in the fourth layer will be established to represent it. The number of FNs in the third and the fourth layers of the MCFIN-B is determined by the similarities between the training patterns and the learning parameters α, β and E.

In chapter 3, we have used a set of fuzzy inference rules to interpret the meaning of the supervised fuzzy inference networks. Fuzzy inference rules can also be used to represent the knowledge acquired by the unsupervised Min-Competitive fuzzy inference network during the training. Now the fuzzy inference rules are in the form of:

rule m: IF: x₁ is A₁, x₂ is A₂, ..., and xₙ is Aₙ
THEN: this pattern belongs to the mth cluster

The proposed MCFIN-A and MCFIN-B can be used to process 2-D visual patterns. The FNs in the first and the second layers are displayed and indexed in two-dimensions. So the proposed MCFIN-A and MCFIN-B have 3-D
structures. However, since the 2-D structure of the first and the second layers only affect the connection weights between the first and the second layers that can be expressed as a function of the indexes of the corresponding FNs, the 3-D structure of MCFINs can be transferred to a 2-D structure in hardware implementation by presenting the index information of the FNs.
5.6 Simulations

MCFIN-A and MCFIN-B were simulated on a 486-PC (33 MHz) using the C language. 36 exemplar patterns (as shown in Fig. 5.6), consisting of 26 English letters and 10 Arabic numerals, have been constructed based on commonly used standard font letters. They are represented in a 16X16 pixel format and have 0 or 1 pixel values. Bold and italic fonts of the standard exemplars (see Fig. 5.7 and Fig. 5.8), and another set of non-standard letters (as shown in Fig. 5.9) were also used to test the performance of the networks.

Each of the 36 exemplar patterns was shifted in eight directions by 1 and 2 pixels respectively. The eight directions are: upward (U), downward (D), left (L), right (R), up-left (UL), up-right (UR), down-left (DL) and down-right (DR). Fig. 5.10 gives all the 16 shifts of the exemplar pattern E. The 36 standard exemplars and some of their shifted versions are used as training patterns. 10 kinds of distorted patterns for each of the 36 letters were also used to test the performances of MCFIN-A and MCFIN-B. The 10 kinds of distortions are: larger (LA), smaller (SM), taller and thinner (TT), squashed (SQ), shaking (SH), disconnected (DC), half-part larger (HL), half-part shifted (HS), added small
parts (AP), missed small parts (MP). Fig. 5.11 gives all the 10 kinds of distortions of four of the 36 exemplar patterns, exemplar C, exemplar H, exemplar Z and exemplar 4.

In the simulation experiments, two sets of patterns that contain the 36 standard exemplars and some of their shifted versions were used as training pattern sets. The patterns chosen in each set are listed in Table 5.1. It should be pointed out that, the training data for MCFIN-A include the patterns and the desired outputs (because MCFIN-A is a supervised network), while the training data for MCFIN-B include only the patterns.

In the simulations, statistical data of the standard exemplars were used to help in making decisions. When the membership values of several classes are too close to make a decision, say the difference is less than $E/2$, then the number of black pixels in the input pattern is compared with the number of black pixels in the prototypes for those classes. The input pattern is assigned to the class whose prototype has the same number of black pixels. If this kind of match does not exist, the input pattern is assigned to the class with the highest membership values or the last found class with the highest membership values.
In the simulations, different values of $\alpha$, $\beta$ and $E_i$ were used for the proposed MCFINs to investigate their effects on the network structures and recognition rates. Tables 5.2 to 5.7 give the simulation results and recognition rates of MCFIN-A. Tables 5.8 to 5.13 give the simulation results and recognition rates of MCFIN-B. In the tables, $M$ is the number of MIN-FNs in the third layer and $P$ is the number of COMP-FNs in the fourth layer.

For illustration, the performances of the MCFIN-A and MCFIN-B are compared to that of the Hamming Network [Lippmann, 1987], which classifies patterns by computing Hamming distances between the input pattern and the prototypes. The recognition results of Hamming Network are summarized in Tables 5.14 to 5.17.
Fig. 5.6 36 standard exemplar patterns
Fig. 5.7 Bold fonts of the 36 standard exemplar patterns
Fig. 5.8 Italic fonts of the 36 standard exemplar patterns
Fig. 5.9 36 non-standard patterns
Fig. 5.10  The shifts of the exemplar pattern E
Fig. 5.11 Distortions of exemplar C, exemplar H, exemplar Z and exemplar 4
Fig. 5.11 Distortions of exemplar C, exemplar H, exemplar Z and exemplar 4

168
### TABLE 5.1  Two sets of training patterns

<table>
<thead>
<tr>
<th>Set name</th>
<th>Patterns included in the Training Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training Set 1</td>
<td>36 standard exemplars</td>
</tr>
<tr>
<td>Training Set 2</td>
<td>36 standard exemplars &amp;</td>
</tr>
<tr>
<td></td>
<td>36 shifted upward by 1 pixel exemplars &amp;</td>
</tr>
<tr>
<td></td>
<td>36 shifted downward by 1 pixel exemplars</td>
</tr>
</tbody>
</table>

### TABLE 5.2  Recognition rates of MCFIN-A when trained by Training Data Set 1 for different fonts of patterns ($\alpha=1.0$, $\beta=0.3$, $E_i=0.1$)

<table>
<thead>
<tr>
<th></th>
<th>Bold</th>
<th>Italic</th>
<th>Non-standard</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
</tbody>
</table>

### TABLE 5.3  Recognition rates of MCFIN-A for shifted patterns ($\alpha=1.0$, $\beta=0.3$, $E_i=0.1$)

<table>
<thead>
<tr>
<th>Training Set</th>
<th>Shift</th>
<th>U</th>
<th>D</th>
<th>L</th>
<th>R</th>
<th>UL</th>
<th>UR</th>
<th>DL</th>
<th>DR</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>97.2</td>
<td></td>
<td>99.7%</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>88.9</td>
<td>83.3</td>
<td>97.2</td>
<td>91.7</td>
<td>97.2</td>
<td>86.1</td>
<td>94.4</td>
<td>86.1</td>
<td>85.4%</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td></td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>100</td>
<td>100</td>
<td>97.2</td>
<td>91.7</td>
<td>97.2</td>
<td>86.1</td>
<td>94.4</td>
<td>86.1</td>
<td>94.1%</td>
</tr>
</tbody>
</table>
### TABLE 5.4 Recognition rates of MCFIN-A for shifted patterns when statistical data of pixel values are also used ($\alpha=1.0$, $\beta=0.3$, $E_t=0.1$)

<table>
<thead>
<tr>
<th>Training Set</th>
<th>Shift</th>
<th>U</th>
<th>D</th>
<th>L</th>
<th>R</th>
<th>UL</th>
<th>UR</th>
<th>DL</th>
<th>DR</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>94.4</td>
<td>94.4</td>
<td>97.2</td>
<td>100</td>
<td>97.2</td>
<td>91.7</td>
<td>94.4</td>
<td>94.4</td>
<td>95.5%</td>
</tr>
</tbody>
</table>

### TABLE 5.5 Recognition rates of MCFIN-A for distorted patterns when trained by Training Data Set 1 ($\alpha=1.0$, $\beta=0.3$, $E_t=0.1$)

<table>
<thead>
<tr>
<th>LA</th>
<th>SM</th>
<th>TT</th>
<th>SQ</th>
<th>SH</th>
<th>DC</th>
<th>HL</th>
<th>HS</th>
<th>AP</th>
<th>MP</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>100%</td>
<td>97.2%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>99.7%</td>
<td></td>
</tr>
</tbody>
</table>

### TABLE 5.6 The simulation results of MCFIN-A when trained by Training Set 2

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>$\beta$</th>
<th>$E_t$</th>
<th>M</th>
<th>P</th>
<th>training time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>0.3</td>
<td>0.1</td>
<td>36</td>
<td>36</td>
<td>137.1</td>
</tr>
<tr>
<td>1.0</td>
<td>0.1</td>
<td>0.1</td>
<td>44</td>
<td>36</td>
<td>123.4</td>
</tr>
<tr>
<td>1.0</td>
<td>0.3</td>
<td>0.1</td>
<td>108</td>
<td>36</td>
<td>147.0</td>
</tr>
<tr>
<td>1.0</td>
<td>0.3</td>
<td>0.2</td>
<td>108</td>
<td>36</td>
<td>147.0</td>
</tr>
<tr>
<td>1.0</td>
<td>0.3</td>
<td>0.3</td>
<td>36</td>
<td>36</td>
<td>137.1</td>
</tr>
<tr>
<td>1.0</td>
<td>1.0</td>
<td>0.1</td>
<td>108</td>
<td>36</td>
<td>151.4</td>
</tr>
<tr>
<td>2.0</td>
<td>0.3</td>
<td>0.1</td>
<td>108</td>
<td>36</td>
<td>147.0</td>
</tr>
</tbody>
</table>
### TABLE 5.7 The recognition rates of MCFIN-A when trained by Training Set 2 for 2 pixel-shifted pattern

<table>
<thead>
<tr>
<th>α</th>
<th>β</th>
<th>E_i</th>
<th>Recognition Rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>U</td>
</tr>
<tr>
<td>0.2</td>
<td>0.3</td>
<td>0.1</td>
<td>88.9</td>
</tr>
<tr>
<td>1.0</td>
<td>0.1</td>
<td>0.1</td>
<td>77.8</td>
</tr>
<tr>
<td>1.0</td>
<td>0.3</td>
<td>0.1</td>
<td>100</td>
</tr>
<tr>
<td>1.0</td>
<td>0.3</td>
<td>0.2</td>
<td>100</td>
</tr>
<tr>
<td>1.0</td>
<td>0.3</td>
<td>0.3</td>
<td>88.9</td>
</tr>
<tr>
<td>1.0</td>
<td>1.0</td>
<td>0.1</td>
<td>100</td>
</tr>
<tr>
<td>2.0</td>
<td>0.3</td>
<td>0.1</td>
<td>100</td>
</tr>
</tbody>
</table>

### TABLE 5.8 Recognition rates of MCFIN-B when trained by Training Data Set 1 for different fonts of patterns (α=1.0, β=0.3, E_i=0.1)

<table>
<thead>
<tr>
<th></th>
<th>Bold</th>
<th>Italic</th>
<th>Non-standard</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
</tbody>
</table>

### TABLE 5.9 Recognition rates of MCFIN-B for shifted patterns (α=1.0, β=0.3, E_i=0.1)

<table>
<thead>
<tr>
<th>Training Set</th>
<th>Shift</th>
<th>U</th>
<th>D</th>
<th>L</th>
<th>R</th>
<th>UL</th>
<th>UR</th>
<th>DL</th>
<th>DR</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>97.2</td>
<td>99.7%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>88.9</td>
<td>83.3</td>
<td>97.2</td>
<td>91.7</td>
<td>97.2</td>
<td>86.1</td>
<td>94.4</td>
<td>86.1</td>
<td>85.4%</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>100</td>
<td>100</td>
<td>97.2</td>
<td>88.9</td>
<td>97.2</td>
<td>86.1</td>
<td>94.4</td>
<td>86.1</td>
<td>93.8%</td>
</tr>
</tbody>
</table>
**TABLE 5.10** Recognition rates of MCFIN-B for shifted patterns when statistical data of pixel values are also used ($\alpha=1.0$, $\beta=0.3$, $E_t=0.1$)

<table>
<thead>
<tr>
<th>Training Set</th>
<th>Shift</th>
<th>U</th>
<th>D</th>
<th>L</th>
<th>R</th>
<th>UL</th>
<th>UR</th>
<th>DL</th>
<th>DR</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>94.4</td>
<td>94.4</td>
<td>97.2</td>
<td>100</td>
<td>97.2</td>
<td>91.7</td>
<td>94.4</td>
<td>94.4</td>
<td>95.5%</td>
</tr>
</tbody>
</table>

**TABLE 5.11** Recognition rates of MCFIN-B for distorted patterns when trained by Training Data Set 1 ($\alpha=1.0$, $\beta=0.3$, $E_t=0.1$)

<table>
<thead>
<tr>
<th>LA</th>
<th>SM</th>
<th>TT</th>
<th>SQ</th>
<th>SH</th>
<th>DC</th>
<th>HL</th>
<th>HS</th>
<th>AP</th>
<th>MP</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>100%</td>
<td>97.2%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>99.7%</td>
</tr>
</tbody>
</table>

**TABLE 5.12** The simulation results of MCFIN-B when trained by Training Set 2

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>$\beta$</th>
<th>$E_t$</th>
<th>M</th>
<th>P</th>
<th>training time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>0.3</td>
<td>0.1</td>
<td>33</td>
<td>33</td>
<td>153.7</td>
</tr>
<tr>
<td>1.0</td>
<td>0.1</td>
<td>0.1</td>
<td>43</td>
<td>43</td>
<td>143.2</td>
</tr>
<tr>
<td>1.0</td>
<td>0.3</td>
<td>0.1</td>
<td>108</td>
<td>108</td>
<td>185.5</td>
</tr>
<tr>
<td>1.0</td>
<td>0.3</td>
<td>0.2</td>
<td>108</td>
<td>108</td>
<td>185.5</td>
</tr>
<tr>
<td>1.0</td>
<td>0.3</td>
<td>0.3</td>
<td>36</td>
<td>36</td>
<td>155.6</td>
</tr>
<tr>
<td>1.0</td>
<td>1.0</td>
<td>0.1</td>
<td>108</td>
<td>108</td>
<td>191.5</td>
</tr>
<tr>
<td>2.0</td>
<td>0.3</td>
<td>0.1</td>
<td>108</td>
<td>108</td>
<td>185.5</td>
</tr>
</tbody>
</table>
TABLE 5.13 The recognition rates of MCFIN-B when trained by Training Set 2 for 2 pixel-shifted pattern

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>$\beta$</th>
<th>$E_t$</th>
<th>Recognition Rate (%)</th>
<th>U</th>
<th>D</th>
<th>L</th>
<th>R</th>
<th>UL</th>
<th>UR</th>
<th>DL</th>
<th>DR</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>0.3</td>
<td>0.1</td>
<td></td>
<td>80.6</td>
<td>75.0</td>
<td>88.9</td>
<td>86.1</td>
<td>77.8</td>
<td>86.1</td>
<td>69.4</td>
<td>77.8</td>
<td>80.2%</td>
</tr>
<tr>
<td>1.0</td>
<td>0.1</td>
<td>0.1</td>
<td></td>
<td>63.9</td>
<td>66.7</td>
<td>69.4</td>
<td>77.8</td>
<td>38.9</td>
<td>22.2</td>
<td>38.9</td>
<td>36.1</td>
<td>51.7%</td>
</tr>
<tr>
<td>1.0</td>
<td>0.3</td>
<td>0.1</td>
<td></td>
<td>100</td>
<td>100</td>
<td>97.2</td>
<td>88.9</td>
<td>97.2</td>
<td>86.1</td>
<td>94.4</td>
<td>86.1</td>
<td>93.8%</td>
</tr>
<tr>
<td>1.0</td>
<td>0.3</td>
<td>0.2</td>
<td></td>
<td>100</td>
<td>100</td>
<td>97.2</td>
<td>88.9</td>
<td>97.2</td>
<td>86.1</td>
<td>94.4</td>
<td>86.1</td>
<td>93.8%</td>
</tr>
<tr>
<td>1.0</td>
<td>0.3</td>
<td>0.3</td>
<td></td>
<td>88.9</td>
<td>83.3</td>
<td>97.2</td>
<td>91.7</td>
<td>75.0</td>
<td>88.9</td>
<td>75.0</td>
<td>83.3</td>
<td>85.4%</td>
</tr>
<tr>
<td>1.0</td>
<td>1.0</td>
<td>0.1</td>
<td></td>
<td>100</td>
<td>100</td>
<td>97.2</td>
<td>100</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>49.7%</td>
</tr>
<tr>
<td>2.0</td>
<td>0.3</td>
<td>0.1</td>
<td></td>
<td>100</td>
<td>100</td>
<td>97.2</td>
<td>100</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>49.7%</td>
</tr>
</tbody>
</table>

TABLE 5.14 Recognition rates of Hamming network when trained by Training Data Set 1 for different fonts of patterns ($\alpha$=1.0, $\beta$=0.3, $E_t$=0.1)

<table>
<thead>
<tr>
<th>Bold</th>
<th>Italic</th>
<th>Non-standard</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>100%</td>
<td>63.9%</td>
<td>75.0%</td>
<td>79.6%</td>
</tr>
</tbody>
</table>

TABLE 5.15 Recognition rates of Hamming network for shifted patterns

<table>
<thead>
<tr>
<th>Training Set</th>
<th>Shift</th>
<th>U</th>
<th>D</th>
<th>L</th>
<th>R</th>
<th>UL</th>
<th>UR</th>
<th>DL</th>
<th>DR</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>33.3</td>
<td>30.6</td>
<td>8.3</td>
<td>5.6</td>
<td>8.3</td>
<td>5.6</td>
<td>11.1</td>
<td>2.8</td>
<td>13.2%</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>19.4</td>
<td>25.0</td>
<td>13.9</td>
<td>11.1</td>
<td>8.3</td>
<td>5.6</td>
<td>8.3</td>
<td>8.3</td>
<td>12.2%</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>100</td>
<td>100</td>
<td>13.9</td>
<td>8.3</td>
<td>13.9</td>
<td>11.1</td>
<td>16.7</td>
<td>11.1</td>
<td>34.4%</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>30.6</td>
<td>27.8</td>
<td>13.9</td>
<td>11.1</td>
<td>5.6</td>
<td>5.6</td>
<td>8.3</td>
<td>8.3</td>
<td>13.9%</td>
</tr>
</tbody>
</table>
TABLE 5.16 Recognition rates of Hamming network for shifted patterns when statistical data of pixel values are also used

<table>
<thead>
<tr>
<th>Training Set</th>
<th>Shift</th>
<th>U</th>
<th>D</th>
<th>L</th>
<th>R</th>
<th>UL</th>
<th>UR</th>
<th>DL</th>
<th>DR</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>58.3</td>
<td>50.0</td>
<td>41.7</td>
<td>36.1</td>
<td>41.7</td>
<td>33.3</td>
<td>47.2</td>
<td>27.8</td>
<td>42.0%</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>50.0</td>
<td>52.8</td>
<td>41.7</td>
<td>33.3</td>
<td>27.8</td>
<td>36.1</td>
<td>33.3</td>
<td>27.8</td>
<td>37.8%</td>
</tr>
</tbody>
</table>

TABLE 5.17 Recognition rates of Hamming network for distorted patterns when trained by Training Set 1

<table>
<thead>
<tr>
<th>LA</th>
<th>SM</th>
<th>TT</th>
<th>SQ</th>
<th>SH</th>
<th>DC</th>
<th>HL</th>
<th>HS</th>
<th>AP</th>
<th>MP</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>30.6</td>
<td>27.8</td>
<td>30.6</td>
<td>36.1</td>
<td>19.7</td>
<td>25.0</td>
<td>55.6</td>
<td>50.0</td>
<td>83.3</td>
<td>58.3</td>
<td>41.4%</td>
</tr>
</tbody>
</table>

From the recognition results of MCFIN-A, we note that the values of $\alpha$, $\alpha$ and $E_\alpha$ affect the recognition rates of the MCFIN-A. The value of $E_\alpha$ should be small to get good recognition results. But small $E_\alpha$ will result in more MIN-FNs in the third layer. The value of $\alpha$ and $\beta$ should be properly chosen. Too small $\alpha$ or $\beta$ value or too large $\alpha$ or $\beta$ value will result in low recognition rates for the MCFIN-A. In the simulations, if $\alpha = 1/(P_{\text{vmax}} - P_{\text{vmin}})$, where $P_{\text{vmax}}$ and $P_{\text{vmin}}$ are, respectively, the maximum pixel value and the minimum pixel value of all the input patterns, and $\beta$ is chosen such that $0.5 = \exp[-\beta(\delta x^2 + \delta y^2)]$, where $\delta x = 2$ and $\delta y = 2$ is the largest shifted pixels in $x$ and $y$ directions, respectively, the resultant MCFIN-A will have a good recognition rate. The recognition rate of
the MCFIN-A becomes worse when the shifts of the patterns are larger. However, the recognition rates can be improved by training the MCFIN-A with some more shifted patterns or using statistical data of the training patterns to help making the decision. The MCFIN-A can recognize almost all of the distorted patterns with 99.7% recognition rate when trained by Training Set 1 with $\alpha=2.0$, $\beta=0.3$ and $E_t=0.1$.

From the learning results of MCFIN-A, we note that if a pattern is similar to one of the prototypes (the maximum output error is less than or equal to $E_t$), the MCFIN-A will treat this pattern as a previously learned pattern without relearning it. If a pattern is not similar to any of the prototypes (the maximum output error is larger than $E_t$), the MCFIN-A will treat this pattern as a new prototype and learn its features. Whether or not a pattern will be treated as a new prototype or as a learned pattern is determined by how much the pattern is similar to the existing prototypes and by the learning parameters $\alpha$, $\beta$ and $E_t$. The values of $\alpha$, $\beta$ and $E_t$ will affect the number of MIN-FNs in the third layer but this number will be larger than or equal to the number of COMP-FNs in the fourth layer and will not exceed the number of training patterns. The structure of MCFIN-A is determined by the training parameters and the training patterns.
From the recognition results of MCFIN-B, we note that if the MCFIN-B cannot separate all the distinct training patterns, the recognition rates are not good. So \( E_t \) should be small enough, \( \alpha \) and \( \beta \) should be large enough to enable the MCFIN-B to separate all the distinct training patterns. However, too small \( \alpha \) or \( \beta \) value or too large \( \alpha \) or \( \beta \) value will result in an MCFIN-B that does not have good recognition rates. Very small \( E_t \) will lead to an MCFIN-B that treats all the different training patterns as prototypes and therefore more FNs are needed in the third and the fourth layer. Consequently, the value of \( \alpha \), \( \beta \) and \( E_t \) cannot be too large or too small. In the experiments, if \( \alpha = 1/(P_{\text{max}}-P_{\text{min}}) \), where \( P_{\text{max}} \) and \( P_{\text{min}} \) are, respectively, the maximum pixel value and the minimum pixel value of all the input patterns, and \( \beta \) is chosen such that \( 0.5 = \exp[-\beta^2(\delta x^2+\delta y^2)] \), the resultant MCFIN-B has a good recognition rate. The recognition rates of the MCFIN-B become worse when the shifts of the patterns are larger. However, the recognition rates can be improved by training the MCFIN-B with some more shifted patterns or using statistical data of the training patterns to help making the decision. The MCFIN-B can recognize almost all of the distorted patterns with a 99.7% recognition rate when trained by Training Set 1 with \( \alpha = 2.0 \), \( \beta = 0.3 \) and \( E_t = 0.1 \).

From the learning results of MCFIN-B, we note that if a pattern that is
similar to one of the existing prototypes (the similarity is larger than 1-\(E_b\)) is fed to the MCFIN-B, the MCFIN-B will treat this pattern as a previously learned pattern without relearning it. If a pattern that is not similar to any of the existing prototypes (all the similarities are less than 1-\(E_b\)) is fed to the MCFIN-B, the MCFIN-B will treat this pattern as a new prototype and learn its features. If the differences between distinct training patterns are small, both \(E_i\) should be small. The values of \(\alpha\) and \(\beta\) will also affect the ability of the MCFIN-B to separate distinct training patterns. If the differences between distinct training patterns are small, \(\alpha\) and \(\beta\) should be large. Whether or not a pattern will be treated as a prototype is determined by how much the pattern is similar to the existing prototypes and by the learning parameters \(\alpha\), \(\beta\) and \(E_i\). So the structure of the MCFIN-B is determined by the training parameters and the training patterns.

From the recognition rates of the MCFIN-A and MCFIN-B, we can see that MCFIN-A is better than MCFIN-B. This is because MCFIN-A is a supervised learning network. It gets more information from training data than the unsupervised learning network MCFIN-B.

From the recognition rates of the Hamming Network, we can see that the performances of the MCFIN-A and MCFIN-B are much better than that of the
Hamming Network. The MCFIN-A and MCFIN-B can recall all the bold, italic and non-standard patterns and almost all of the distorted patterns while the Hamming Network cannot. The Hamming network works well only for bold patterns and three kinds of distorted patterns. The recognition rates of the Hamming Network for shifted patterns and other distorted patterns are not good.

5.7 Conclusions

In this chapter, four-layer feedforward fuzzy inference networks for invariant pattern recognition problems, a supervised MCFIN and an unsupervised MCFIN, are proposed with their associated learning algorithms. The proposed MCFIN-A and MCFIN-B have neural network structures and learning ability while using fuzzy inference algorithms to process 2-D visual patterns. Feature extraction is incorporated in the structures of the networks. They can recognize patterns without external assistance. The learning speed of the proposed MCFIN-A and MCFIN-B is much faster than that of any neural network using the backpropagation algorithm. The proposed MCFIN-A and MCFIN-B perform extremely well when used in pattern recognition problems, even when the input patterns are shifted or distorted. The recognition rates of the MCFIN-A and MCFIN-B are much better than those of the Hamming network.
CHAPTER VI

CONCLUSIONS AND SUGGESTIONS FOR FUTURE WORK

In this dissertation, the definition of the artificial neurons has been generalized. Fuzzy neurons have been defined and five types of fuzzy neurons have been introduced. Fuzzy inference methods for pattern recognition have been developed. These inference methods are similar to the inference methods for control systems but are particularly suitable for pattern recognition systems. Fuzzy inference networks for pattern recognition problems have been presented. The structures and learning algorithms of fuzzy inference networks based on the proposed inference methods have been developed.

Two of the proposed fuzzy inference networks (Min-Max Fuzzy Inference Network and Min-Sum Fuzzy Inference Network) have been applied to pattern classification problems. These two FIN classifiers can learn from fuzzy labelled or nonfuzzy labelled data and find soft and hard partitions. The learning and classification speed of these FINs is very fast. The classification results of these two FINs are better than those of the Fuzzy ARTMAP [Carpenter et al., 1993].

179
These two FIN classifiers can be used in the applications where an intelligent decision has to be made on classes with overlapping characteristics. They can be used in fuzzy control problems, and other fuzzy knowledge-base systems where it is required to learn fuzzy inference rules.

Fuzzy inference networks based on the Min-Competitive inference method have also been developed for letter recognition systems. There is a supervised learning Min-Competitive fuzzy inference network and an unsupervised learning Min-Competitive fuzzy inference network. These two fuzzy inference networks have been constructed in particular for 2-D visual pattern recognition problems and have been tested with letter patterns with black and white pixel values. The feature extraction is incorporated in the structures of the networks. These two fuzzy inference networks are self-organized during the learning procedure. The learning speed is much faster than those neural networks using backpropagation algorithms. The structures of these two fuzzy inference networks are simple and their recognition speeds are fast. These two fuzzy inference networks perform extremely well when used in pattern recognition problems, even when the input patterns are shifted or distorted. The recognition rates of these two fuzzy inference networks are much better than those of the Hamming network.
The fuzzy inference networks proposed in this research are fuzzy systems with neural network structures. They possess parallel structures and learning ability. During the learning procedure, the membership functions and fuzzy inference rules for fuzzy systems can be automatically determined. The knowledge represented by the networks can be interpreted by the fuzzy inference rules obtained by the networks. Consequently, the proposed fuzzy inference networks have combined the advantages of neural networks and fuzzy systems and have avoided the disadvantages of both.

In this research, the author has

(1) Generalized the definition of a neuron into a fuzzy neuron;
(2) Developed fuzzy inference methods for pattern recognition problems;
(3) Implemented fuzzy inference networks based on the developed fuzzy inference methods and fuzzy neurons;
(4) Developed self-organizing learning algorithms for the proposed fuzzy inference networks;
(5) Applied the proposed fuzzy inference networks to pattern classification problems and invariant pattern recognition problems.
The future work of this research could be:

1. To investigate new network structures using the proposed fuzzy neurons;

2. To construct large intelligent pattern recognition systems by using the proposed fuzzy inference networks as sub-systems and to develop the associated learning algorithms;

3. To apply the proposed fuzzy inference networks to other related areas, such as fuzzy control systems and system modelling.

The success of the proposed fuzzy inference networks in pattern classification and recognition problems demonstrate that the attempt to establish fuzzy inference networks by combining the strengths of fuzzy logic and neural networks is fruitful.
REFERENCES


pp. 461-465.


[28] Grossberg, S., "How does a brain build a cognitive code?"


[74] Sanchez-Sinencio, E. and Lau, C., *Artificial Neural Networks, Paradigms, and Hardware Implementations*, IEEE Press, Piscataway,


VITA AUCTORIS

NAME: Yaling Cai

PLACE OF BIRTH: Zunyi, P. R. China

YEAR OF BIRTH: 1963

EDUCATION: Department of Electrical Engineering
Tsinghua University, Beijing, China
1980-1985, B. E.

The Second Institute of Aerospace Ministry
Beijing, China
1985-1988, M. E.

Department of Electrical Engineering
University of Windsor, Windsor
Ontario, Canada
1991-1995, Ph.D.