An Enhanced Progressive Fuzzy Clustering Approach to Pattern Recognition

A thesis submitted for the degree of
Doctor of Philosophy

Paul Poh Teng IM

June 1997

Department of Electrical and Electronic Engineering
Faculty of Engineering
Victoria University of Technology
P.O. Box 14428, MCMC, Melbourne
Victoria 8001, Australia

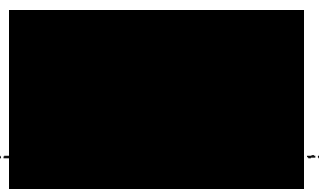
Declaration

This thesis contains the work of my doctoral studies under the supervision of Mr Michael Wingate (Victoria University of Technology), Dr Wee Sit Lee (Victoria University of Technology) and Dr Bin Qiu (Monash University). The contents of this thesis have been referenced in the following papers that have either been published at conference proceedings or are under review for journal publications:

1. Im P.T., Qiu B. and Pleasants A. (1993). Restoration of degraded segmented images using neural networks, *2nd Conference on Digital Image Computing: Techniques and Applications*, Macquarie University, Sydney, pp. 579-586.
2. Qiu B., Im P.T. and Pleasants A. (1994). The design of a neural network configuration for object recognition, Vol. II, *IEEE International Conference on Acoustics, Speech and Signal Processing*, Adelaide, pp. 561-564.
3. Im P.T., Qiu B., Wingate M. and Pleasants A. (1994). Autonomous image segmentation using fuzzy *c* means and neural network techniques, *3rd International Conference on Automation, Robotics and Computer Vision*, Singapore, pp. 920-924.
4. Qiu B. and Im P.T. (1995). Pattern classification for real world image using fuzzy *c* means and neural network methods, *IEEE International Conference on Neural Networks and Signal Processing*, Nanjing, China, pp. 365-368.
5. Im P.T. and Qiu B. (1995) A neural network implementation for fuzzy clustering of images, *IEEE International Conference on Neural Networks and Signal Processing*, Nanjing, China, pp. 254-257.
6. Im P.T., Wingate M. and Qiu B. (1995). A cluster prototype centring by membership algorithm for fuzzy clustering, *2nd Asian Conference on Computer Vision*, Vol. 2, Singapore, pp. 67-70.
7. Im P.T., Qiu B., Wingate M. and Herron L. (1995). Linear boundary detection by cluster prototype centring based on fuzzy memberships, Vol. I, *Australia New Zealand Intelligent Information Processing Conference*, Perth, Australia, pp. 210-212.
8. Im P.T., Qiu B., Wingate M. and Herron L. (1995). Implementing a neural network for a progressive fuzzy clustering algorithm, Vol. III, *IEEE International Conference on Neural Networks*, Perth, Australia, pp. 1369-1372.
9. Im P.T. (1995). A fuzzy neural technique for object recognition, *3rd Conference on Digital Image Computing: Techniques and Application*, Brisbane, Australia, pp. 259-263.
10. Im P.T., Lee W.S. and Wingate, M. (1996). A progressive fuzzy clustering approach for the detection of linear boundary and corners, *Australian Journal of Intelligent Information Processing Systems*, Vol. 3, No. 3, pp. 41-58.
11. Bezdek J.C. and Im P.T. (1996). A new infinite family of generalized *c*-means partitioning models, *IEEE Transactions on Fuzzy Systems*. (submitted)

Statement of Originality

This thesis contains no material which has been accepted for the award of any other degree or diploma in any university and that, to the best of the author's knowledge and belief, the thesis contains no material previously published or written by any other person, except where reference is made in the text of the thesis.



Paul Poh Teng IM
Department of Electrical and Electronic Engineering
Victoria University of Technology
P.O. Box 14428, MCMC
Melbourne, Victoria 8001
AUSTRALIA

Acknowledgments

I have been most fortunate to enjoy excellent supervision of my studies from Mr Michael Wingate, Dr Bin Qiu and Dr Wee Sit Lee. The three short years under their skilful academic custodianships have left me with an enduring impression of their caring attitude, support and professionalism.

I am grateful to the Victoria University of Technology for the healthy research environment provided by the Department of Electrical and Electronic Engineering, which has been quite conducive to my research progress. The university's library staff have provided an efficient service to my numerous request for research publications. The department gave me generous tutoring hours to hone programming and teaching skills. There are staff at the department for whom special thanks are due. Department secretaries, Mrs Shirley Herrewyn and Mrs Somchit Panitsiri have facilitated my journey through the research environment. The technical staff, Mr Ralph Phillips, Mr Zoltan Varga, Mr Brian Healey and Mr Les Nakonieczny, have made my experimental tasks less of a burden. Associate Professor Wallace Evans, former Head of the Electrical and Electronic Engineering Department, has been supportive of this research and funded presentation of papers at interstate conferences. I am grateful to Associate Professor Leonard Herron (recently retired), whose powerful recommendation enabled me to receive a highly competitive PhD scholarship, and thus the freedom to do research undistracted by financial concerns.

Interaction with academic staff of the department has provided stimulating ideas and rich source of materials for research. Mrs Ann Pleasants and Associate Professor Len Herron provided a helpful review of topics selected for conference proceedings. Dr Gregory Cain, Mr Alexander Simcock and Dr Leon Reznik have made helpful contributions to neural networks and fuzzy theories. Mr Michael Wingate, an expert on pattern recognition, has made a substantial contribution on this topic. Five distinguished

academic colleagues, Mike Wingate, Wee Sit Lee, Bin Qiu, Ann Pleasants and Len Herron have collaborated with me on several papers. It has also been a happy research experience working with Mr Robert Ives, Dr Larissa Al-Dabbagh, Dr Juan Shi, Dr Dirk Chiu, Dr Hao Shi, Mr Daniel Nelson, Mr Yau Man Ng, Mr Jack Singh, Mr Malcolm Dow, Mr John Chlond, Mrs Elizabeth Haywood, Dr Roman Malyniak and Associate Professors Aktar Kalam and Patrick Leung (new department Head of Electrical and Electronic Engineering).

I wish to thank Professor Geoffrey Lleonart, from the Department of Mechanical Engineering, and the Research Centre for Packaging and Handling, for the invaluable experience as a Research Assistant.

Material contributions to the research were made by the following people. Mr Nick Sokolov, from the Mechanical Engineering Department, provided the wool samples for the images of Figures 4.1 and 4.2 in Chapter 4. The images of Figures 5.5, 5.6, 5.7 and 5.9 in Chapter 5 were supplied by Mike Wingate from the Department of Electrical and Electronic Engineering. Mr Terence English, Geelong proprietor of EVTEK Computer Systems, provided the software for capturing screen images displayed in the thesis. Images of the tile surface defect patterns of Figures 4.3 and 4.4 in Chapter 4 are obtained from the roof tiles donated for research by Color Vision Systems Pty. Ltd., Victoria.

I am greatly indebted to Mike Wingate and Wee Sit Lee. They critically reviewed draft versions and made numerous invaluable suggestions reflected in the thesis.

This research is inspired by Professor James Bezdek (Dept of Computer Science, University of West Florida, USA), an internationally recognised fuzzy theorist and founder of the *Fuzzy c-Means* and modern fuzzy clustering algorithms. Our modest research collaboration on the *Possibilistic Fuzzy c-Means* has shaped my understanding of the profundity of fuzzy clustering for pattern recognition.

I am thankful to Sian, my lovely wife, whose patience and understanding have contributed in subtle ways to the successful accomplisment of the thesis.

Abstract

This thesis applies an enhanced progressive clustering approach, involving fuzzy clustering algorithms and fuzzy neural networks, to solve some practical problems of pattern recognition. A new fuzzy clustering framework, referred to as Cluster Prototype Centring by Membership (CPCM), has been developed. A Possibilistic Fuzzy c-Means algorithm (PFCM), which is also new, has been formulated to investigate properties of fuzzy clustering. PFCM extends the useability of the Fuzzy c-Means (FCM) algorithm by generalisation of the membership function.

CPCM provides a flexible framework to integrate clustering methods that detect cluster substructures. Four pattern recognition theories, consisting of the Bayes decision rule, partitional clustering, fuzzy clustering and neural network, which influenced the development of the CPCM clustering model and application algorithms are reviewed. Four new experimental algorithms to detect compact cluster regions and outlines are adduced to illustrate the adaptation of analytic fuzzy clustering algorithms for the CPCM framework. Three new cluster validity indices are developed to evaluate the clustering performance of the basic k-Nearest Neighbour, FCM, PFCM and CPCM based algorithms.

Application development is focused on three problem contexts: (i) detection of contaminants in wool and paint defect on tile surface (region segmentation), (ii) identification of real object lines and circles (boundary detection) and (iii) recognition of a notched feature on an armature housing (general pattern recognition). The CPCM algorithms demonstrate more accurate segmentation of small scale defect patterns compared to FCM. Results obtained from these algorithms indicate robust clustering and accurate identification of cluster parameters (circle centre, radius, line gradient and corners) from real data silhouettes characterised by the presence of noise, fragmentation and partial obscurity. These algorithms also facilitate a solution for general pattern recognition. Several fuzzy neural network configurations are developed to improve object recognition and to model the cluster prototype from a progressive clustering algorithm.

List of Figures

Figure — Page

Figure Page

Figure Page

List of Tables

Table Page

Table Page

List of Principal Symbols

Symbol — Section

The following definitions of symbols apply in the thesis except where otherwise indicated.

$\lvert \cdot \rvert$	Absolute value	2.2
$\lVert \cdot \rVert$	Norm or length of a vector	2.4.8.2
$\lVert \cdot \rVert_A$	Weighted norm with symmetric positive definite matix A	2.4.4
{...}	Set description by list	2.4.4
[a,b]	Closed real-valued interval, $a \leq x \leq b$	2.4.4
$N(\mu,\Sigma)$	Multivariate normal density with mean μ and covariance Σ	2.2
$p(x\|w_j)$	Probability density function of random variable x given class w_j	2.2
$P(w_j\|x)$	A posteriori probability of class w_j given random variable x	2.2
$P(w_j)$	A prior probability of class w_j	2.2
$m \to k^-$	m approaches k from above	2.4.5
$p \Rightarrow q$	p implies q	2.4.4
$p \Leftrightarrow q$	p if and only if q	8.2.3.2
$\cap$	Intersection	2.3
$\cup$	Union	2.3
$\in$	Belongs to	2.4.4
$\tilde{A}$	Complement of set A	2.4.4
$\varnothing$	Empty or null set	2.4.4
$\{x \mid p(x)\}$	Set description by property	2.4.9
$h : X \to Y$	Function h maps X into Y	2.4.6.2
$\Re^d$	d dimensional space over reals	2.1.5
∞	Infinity	2.4.4
φ	Neural network learning rate parameter	2.5.2
max	Maximum	2.1.5
min	Minimum	3.3
$d(i,j)$	Distance between two points i and j	3.3

Symbol		Section
$s(k)$	Similarity index of point k	3.3
$\bar{s}_1(j)$	Similarity index of jth cluster	3.3
$\bar{s}_2$	Similarity index of data set	3.3
$\mathbf{x}_k$	kth feature vector or data item	2.4.4
$\mathbf{v}_j$	jth cluster or class prototype	2.4.4
α	PFCM exponent on membership constraint	2.4.5
α_t	Alphacut, a membership threshold	2.4.7.1
c	Number of clusters, partitions or classes	2.4.4
δ, ε	Termination condition for iterative procedures	3.4.3
d_{ik}	Distance of feature point $\mathbf{x}_k$ from prototype $\mathbf{v}_i$	2.4.4
d_k	Distance of feature point $\mathbf{x}_k$ from prototype $\mathbf{v}$ in CPCM	3.4.1
q	Fuzzifier, a CPCM exponent on d_k	3.2.1
p	PFCM exponent on d_{ik}	2.4.5
r_{min}	Minimum cluster radius	3.4.1
r_k	Fixed cluster radius in CPCM	3.4.1
η_j	The jth intra-cluster distance	2.4.7.2
η_{min}	Minimum η corresponding to r_{min}	3.4.1
u_{ik}	Membership of $\mathbf{x}_k$ in the ith prototype	2.4.4
u_k	Membership of $\mathbf{x}_k$ in CPCM	3.4.1
m	Fuzzifier or smoothing exponent on membership	2.4.4
k	Index of data item or feature vector	2.4
N	Number of data items	2.4.4
N_j	Number of points in the jth cluster	3.4.1
N_α	Number of points in cluster with $u_{ik} > \alpha_t$	3.2.3.2
N_c	Number of points in current data	3.2.3.2
N_{min}	Minimum points specification for a valid cluster	3.2.3.2
f_c	Cohesion factor	3.4.1
X	Data set of unlabelled N feature vectors	2.4.4
U	$c \times N$ membership matrix, a fuzzy c-partition of X	2.4.4
V	Set of c-tuples of cluster prototypes in the fuzzy c-partitions of X	2.4.4
J_m	Objective function of fuzzy clustering algorithm	2.4.4

Symbol		Section
lgap	Specification gap separating two lines	5.3
pgap	Specification gap between two ordered adjacent pixels	5.3
egap	Specification gap between two nearest ends of lines	5.3
cgap	Specification line extension gap to form a corner	5.3
t	Iteration index	3.4.1
ln	Natural logarithm	2.2

The following is a list of abbreviations used in the thesis.

KNN	*k*-Nearest Neighbour algorithm	2.3.3
FCM	Fuzzy *c*-Means algorithm	2.4.4
HCM	Hard *c*-Means algorithm	2.4.3
PFCM	Possibilistic Fuzzy *c*-Means algorithm	2.4.5
PCM	Possibilistic *c*-Means algorithm	2.4.7.1
EPCM	Enhanced Possibilistic *c*-Means algorithm	2.4.7.2
FKR	Fuzzy *K*-Rings algorithm	2.4.8.1
FKE	Fuzzy *K*-Ellipse algorithm	2.4.8.2
CPCM	Cluster Prototype Centring by Membership algorithm	3.2.3
SFM	Sequential Fuzzy Means	3.2.3.1
AFC	Adaptive Fuzzy Clustering	5.1
HT	Hough Transform	5.1
CA	Centre Approximating	6.2.3
OCF	Optimal Circle Fit	6.2.2
LSE	Least Squares Error	2.3.2
DM	Direct Mapping	8.1.2
SFPM	Scaled Fuzzy Prototype Mapping	8.1.3
EWR	Effective Width Ratios	8.1.4
SOM	Kohonen Self Organising Feature Maps	2.5.2
FKCN	Fuzzy Kohonen Clustering Network	2.5.2
FFBP	Feed Forward BackPropagation	2.5.3
MAE	Maximum Absolute Error	8.3.4
ISODATA	Iterative Self Organising Data Type A	2.4.4

Contents

Chapter 1

Introduction

This thesis is about the understanding and interpretation of images acquired from a video camera. Humans perform this task almost instantly by intuition. It is however, incredibly difficult to replicate this task on a machine. This is because the *articulated* aspects of human perception are incongruent with the *knowing* aspects. Consequently, the goals of pattern recognition must be considered *idealisations* of the real world conditions and the results of pattern recognition methods, *approximations* of human perceptions. By limiting the conditions upon which the graphical image of the object is dependent, some useful results can be obtained. Indeed, the recognition of these limitations has produced some successful applications. Notably, in the areas of large scale integrated electronics, automotive manufacturing, ecology, meteorology, brain scan diagnostics and vision guided robotics.

In the thesis, a new enhanced progressive fuzzy clustering approach is applied to solve some practical problems in the following areas of pattern recognition: (i) region segmentation, (ii) boundary detection and (iii) general pattern recognition. New fuzzy clustering algorithms have been developed to enable successful and accurate identification of cluster parameters in a noisy environment. New fuzzy neural configurations have been designed to enable accurate pattern classification for object recognition and cluster prototype modelling.

Section 1.1 of this chapter, introduces the background information and motivations for pattern recognition. This is followed by a discussion of the scope and contributions of the thesis in Section 1.2. The last section of this chapter gives an overview of each chapter of the thesis.

1.1 Background and Motivations

This thesis is primarily concerned with the use of fuzzy clustering, specifically the Cluster Prototype Centring by Membership (CPCM) framework, to solve practical problems of **pattern recognition** such as the detection of defect patterns or identification of structure in **data**. Fuzzy clustering may be broadly defined as a cluster analysis or neural network technique that applies fuzzy concepts based on fuzzy set theories to partition data.

Cluster analysis is the exploration of data for structure or clusters. **Clusters** are natural groupings or **partitions** in data. Cluster analysis organises data that are typically not **labelled** by a process which attempts to disclose the **structure** or geometric properties in data. By organising data into discrete groups, cluster analysis performs **classification**. The study of data groupings has a long history involving diverse disciplines and known under various names such as cluster analysis, numerical taxonomy and automatic data classification. Cluster analysis is widely used in many different fields including the following: artificial intelligence, vision guided robotics, medical research, remote sensing, biology, psychology, voice recognition, political science, economics, meteorology and ecology.

1.1.1 Background

A representative context of **image processing** for pattern recognition is shown in Figure 1.1. The pattern recognition applications presented in the thesis, from Chapters 4 to 7, assume this context for data **preprocessing** and may also involve the use of a knowledge or rule base within specific algorithms for **robust** clustering.

The problem domain is a problem to be solved by a machine vision system or **computer vision** such as the detection of contaminants in wool samples or to count the number of cells in a blood sample. The image acquisition and preprocessing stages are also known as low level stages. These involve the acquisition of information from the world using a camera or other transducers, data conversion to a form suitable for computer processing, and preprocessing to improve the image for subsequent or downstream stages. Preprocessing typically involves techniques for enhancing contrast, removing **noise**, and isolating **regions**. The intermediate level stages include **segmentation** and representation or **feature** description. Segmentation may be broadly defined as the partitioning of an **image** into its

constituent parts or objects. In isolating an object from a scene, segmentation reduces the level of complexity and the amount of data for processing and so simplifies the task for object feature extraction. Feature extraction deals with extracting features or **patterns** that result in some quantitative information of interest or features that are basic to discriminating one class of objects from another. Pattern recognition and **interpretation** are considered a high level stage of image processing. Note that pattern recognition is used in a wider sense to include object recognition. Pattern recognition is a process that assigns a label to an object based on the information provided by its descriptors. Interpretation involves assigning meaning to an ensemble of recognised objects. Knowledge about the problem domain is coded into the image processing system as a knowledge base. The knowledge base interacts with each stage to facilitate the process of pattern recognition. Some fuzzy algorithms incorporates the lower and mid-level stages within the same algorithm. In Chapter 7 an example is given to illustrate the use of the image processing stages shown on Fig. 1.1.

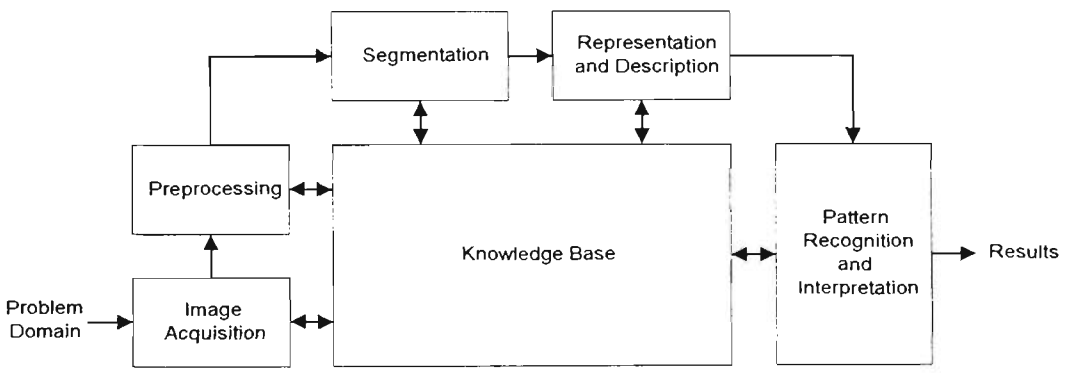


Figure 1.1 Basic steps in image processing. Adapted from [Gonzalez and Woods, 1992].

An object recognition system finds objects in the real world from object models which are known a priori. The problem can be described as assigning a set of labels to an image containing one or more objects of interest, corresponding to a set of models known to the system. Another way of representing the basic steps to object recognition is shown in Fig. 1.2.

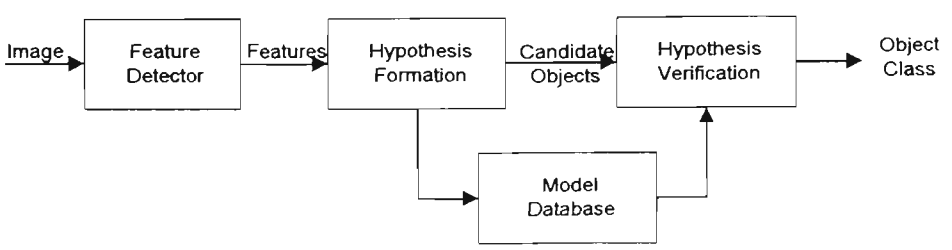


Figure 1.2 Basic steps to object recognition. Adapted from [Jain et al., 1995].

The model database contains all the representative models to identify images presented for object recognition. Information in the model database depends on the problem domain, and the methods of recognition. The feature detector operates on the image to assist with object hypothesis. The features selected depend on the objects to be recognised and the organisation of the model database. From the detected features, the hypothesiser assigns likelihoods to objects in the scene to facilitate a solution by limiting the search space. The model database uses an indexing scheme to eliminate unlikely candidate objects from recognition consideration. The hypothesis formation and verification components vary in importance depending on the recognition methods used. For example, the general pattern recognition scheme in Chapter 7 does not require hypothesis formation.

Some of the important issues to be considered in the design of an object recognition system are:

- *Object or model representation*: How should objects be represented in the database? What important object features are to be captured by the models?
- *Feature extraction*: Which features are to be detected? How reliably and with what degree of **accuracy** can they be detected? Are they adequate to enable object **identification**?
- *Feature model matching*: How to match features in images to the models in the database? An exhaustive match in many cases may be too slow for real-time applications. Effectiveness of features and matching efficiency must be considered in developing a pattern matching method.
- *Hypothesis formation*: How can a set of likely objects based on feature matching be selected? How to assign probabilities to each possible object? This step uses heuristics to reduce the size of the search space.
- *Object verification*: How can object models be used to select the most probable object from an image? The models provide a means of verifying the probability of a correct decision.

Object recognition is a complex process involving many factors that determine the result of recognition. Some of these factors are:

- *Scene constancy*: Are the scenes similar to the conditions of the models? The scene may contain problems for recognition such as **illumination** variations, shadows, camera pa-

rameters and camera viewpoint. Multiple objects can present problems for recognition, by touching or overlapping one another, or casting shadows on other pieces of objects.

- *Image-models spaces*: Three dimensional objects tend to be too complex to model. If these objects can be approximated by a 2D model, the problem of perspective effects can be greatly reduced and model representation considerably simplified.
- *Number of objects in the model database*: If the number of objects is small, the hypothesis formation stage may not be necessary. A large number of objects has implications for increasing computation complexity and feature selection.

1.1.2 Motivations

A significant reason to use cluster analysis is related to the problem of data size. To illustrate, Jain and Dubes [1988] showed that a *brute force* approach to the partitioning of data is quite impractical, even for a small number of partitions. If *S(N,K)* denotes the number of clusterings of N objects into K clusters, the solution is the Stirling number of the second kind given by

$$S(N,K)=\frac{1}{K!}\sum_{i=1}^{K}(-1)^{K-i}\binom{K}{i}i^{N}$$

There are 34,105 distinct partitions of 10 objects into 4 clusters, but grows explosively to 11,259,666,950 partitions for 19 objects and 4 clusters. Cluster algorithms can significantly alleviate this kind of problems and consequently, reduce the time and effort to analyse data.

Academic interest is certainly a strong motivation for this research. Automatic classification is still a very new science, undergoing a vigorous but exciting growth. The diversity of algorithms is an indication that no general definition of a cluster exists. In particular, this research attempts to improve the utility of fuzzy clustering techniques by the inclusion of features that automatically find the number of clusters in data, enhance computation efficiency, accuracy of clustering and the algorithm's ease of use. At a pragmatic level, there are potentially immense rewards to be reaped from this technology. Activities that are ripe for the application of this technology are those conducted in environments considered inimical to human health and life, of a boring, stressful or fatiguing nature and for processes which exceed human capabilities.

1.2 Scope and Contributions

1.2.1 Scope

Recent research publications [see Chapter 6 of Yager and Zadeh, 1992; Krishnapuram and Keller, 1992] indicate firstly, that fuzzy clustering provides a general approach to pattern recognition and secondly, it is able to characterise complicated data substructures [Krishnapuram et al., 1995]. For the purpose of the thesis, the algorithms are restricted to **gray scale** patterns in 256 × 256 **pixel** resolution. This has been done to manage, program development and testing, the efficient use of computer memory and processing time. For other applications, these algorithms may be generalised to higher dimensions.

This thesis is presented in roughly two parts, theory and applications. The first part, comprising Chapters 1 and 2, describes the context, basic concepts and theories of pattern recognition. The second part, from Chapters 3 to 8, examines the practical applications of the fuzzy clustering algorithms developed from the pattern recognition theories of Chapter 2. Applications are limited to the following three categories: (i) region segmentation, (ii) boundary detection and (iii) general pattern recognition. The fuzzy clustering equations introduced in the application algorithms of Chapters 3 to 8 freely assume the pattern recognition theories given in the first two chapters, without further elaboration.

1.2.2 Contributions

My contributions to the field of pattern recognition can be considered at two levels. The first level relates to the algorithms' basic characteristics, their significant design and performance features. The second level considers the application aspects of these algorithms and the kind of problems they can effectively solve.

1.2.2.1 First Level Contributions

The first level contributions may be summarised by the following seven points:

- Possibilistic Fuzzy *c*-Means (PFCM) extends the membership functions of the Fuzzy *c*-Means (FCM) to enhance the selectivity, controllability and nature of clusters.
- Sequential Fuzzy Clustering (SFM) with a single prototype, as an alternative to FCM.

- Enhanced Possibilistic *c*-Means (EPCM) improves clustering at local centroids.
- Cluster Prototype Centring by Membership (CPCM) finds the number of clusters in the data set automatically without needing cluster validity testing like FCM.
- CPCM significantly improves clustering speed by the progressive removal of cluster points or noise points, compared to FCM.
- CPCM detects small scale defect problems more accurately compared to FCM.
- Three cluster validity indices for the objective evaluation of clustering performance.

1.2.2.2 Second Level Contributions

The second level contributions may be summarised under the following seven points:

- Fuzzy neural network applied to improve object recognition by statistical correlation coefficient matching.
- Fuzzy neural network applied to improve cluster identification under various illuminations.
- Fuzzy neural network applied to model fuzzy cluster prototypes.
- CPCM applied to progressive clustering based on the Fuzzy *c*-Means.
- CPCM applied to region segmentation.
- CPCM applied to linear and circular boundary detection.
- CPCM and pattern matching algorithms applied to general pattern recognition.

1.3 Overview of Thesis

Chapter 1 outlines the background and motivations for this research. It describes the scope and contributions of the thesis.

Chapter 2 introduces four theories for pattern recognition comprising: (i) Bayes decision rule, (ii) partitional clustering, (iii) fuzzy clustering and (iv) neural networks. The Bayes decision rule introduces fundamental concepts of classification and the discriminant function. Partitional clustering theory is used to derive the *k*-Nearest Neighbour algorithm and to illustrate cluster concepts and properties. The basic structure of FCM is discussed, followed by Bezdek and Im's PFCM, developed to extend FCM clustering properties. The variable norms of Gustafson and Kessel's algorithm, and Gath and Geva's algorithm are examined. The Possibilistic *c*-Means (PCM) algorithm from Krishnapuram and Keller is

reviewed. Bezdek and Im's EPCM algorithm is developed to improve PCM's clustering performance at local centroids. The clustering performance of FCM, PFCM, PCM and EPCM clustering algorithms are evaluated. The basic architecture and learning characteristic of a backpropagation neural network are presented. Two different ways of designing a fuzzy neural network, by model embedment and by model mapping are discussed.

Chapter 3 examines four experimental algorithms to illustrate the adaptation of analytic fuzzy clustering algorithms for the CPCM framework. The experimental algorithms explore a variety of geometric cluster structures. SFM is developed to explore an alternative scheme for progressive clustering. Comparative tests of clustering performance are conducted on the FCM, KNN and CPCM algorithms.

Chapter 4 applies the CPCM framework to the segmentation of regions. Fuzzy clustering algorithms are developed to solve pattern recognition problems relating to: (i) detection of contaminants in wool samples and (ii) detection of surface defects in roof tiles. The CPCM based clustering algorithm demonstrates superior detection of small scale defect patterns compared to FCM.

The detection of linear boundaries is presented in Chapter 5. This and the detection of circular boundaries in Chapter 6, demonstrate useful extension of the basic FCM algorithm and the accurate identification of cluster parameters in the presence of noise.

Chapter 7 describes a solution to a general pattern recognition problem involving a combination of pattern matching and fuzzy clustering algorithms. It demonstrates how fuzzy clustering techniques facilitate the detection of an arbitrary feature from the top face of an armature housing object.

Chapter 8 presents three neural network applications to: (i) improve object recognition, (ii) improve cluster identification and (iii) extract cluster prototypes from image data. The fuzzy neural networks demonstrate improved classification performance compared to the conventional neural networks.

Chapter 9 concludes the thesis with a summary of the major conclusions and suggestions for future research directions.

Chapter 2

Methods of Pattern Recognition

Basic concepts of image processing and cluster analysis are introduced in Section 2.1 to prepare the context for a review of the theories of pattern recognition in the next four sections. The Bayes decision theory is presented in Section 2.2, partitional clustering theory in Section 2.3, fuzzy clustering theory in Section 2.4 and neural network theory in Section 2.5. The construction of fuzzy clustering models and algorithms in Chapters 3 to 8 assumes to a large extent the pattern recognition theories of this chapter, particularly fuzzy clustering theories and methods from Section 2.4.

The materials of this chapter involve quite extensive use of differential calculus because of the optimisation problems associated with objective functions. The contents are intended to be a succinct but fairly comprehensive survey of the field, directed at fuzzy clustering practitioners. This chapter may be skipped to enjoy the practical flavour of the applications and returned to later for advanced insight into the pattern recognition theories.

The discussion of fuzzy clustering theories is largely expressed through the properties of vectors in real space. To define the usage of vectors in the thesis and to provide sufficient details to arrive at correct results, Appendix A is included. It reflects the convention of the leading FCM practitioners.

2.1 Introduction

Duda and Hart, in a preface to their book [1973, p. vii], define pattern recognition as the "machine recognition of meaningful regularities (of data) in noisy or complex environ-

ment". This definition makes two important statements for pattern recognition: (i) useful data is significant and has meaningful patterns in contrast to random patterns and (ii) the data is typically embedded in noise or other complex features.

2.1.1 Nature of Data

Data in the context of the thesis, consist mostly of **digital images** acquired from a video camera. However, the clustering algorithms place no restriction on the type of data for clustering provided it can be represented symbolically. Video signals can be acquired in either gray tones or typically in the three primary colors of red, green and blue (in video systems, there are several other image formats). Processing color signals requires more computer memory and intensive processing for each color component and sometimes mixture of components. Therefore for the purposes of the research, images in 256 levels of gray, at a resolution of 256 × 256 (row × col) pixels are used. This specification gives adequate feature resolution without excessive demands on processing power from personal computer systems. Another reason for gray scale image processing is that it is not difficult to make the transition to color processing.

2.1.2 Data Preprocessing

Next, we discuss part (ii) of Duda and Hart's definition of pattern recognition. It has been previously noted that real data is typically embedded in noise or a complex environment. Data for analysis may be associated with one or more objects of interest. To enable analysis of the object in data, it is first necessary to separate the objects from the environment or background. This process is called image segmentation. It is possible to apply clustering algorithms that operate directly on raw image data (see Chapter 4) but in some cases, (see Chapter 7) it is necessary to process the data before applying the clustering algorithm. One common process (among several others) uses a technique called **thresholding**. Thresholding uses an image **point operator** to produce a **binary image** from a gray scale image. Mathematically, any point function $f(x,y)$, for which $f(x,y) > T$ where T is the intensity or color threshold, is regarded as an object point with some value greater than zero. The background is assigned a value of zero. Bi-level thresholding segments an object in the range of $T_1 \leq f(x,y) \leq T_2$. Methods for image thresholding are quite diverse of which a sample is given in [Sonka et al., 1993; Davies, 1990; Haralick and Shapiro, 1992; Gon-

zalez and Woods, 1992]. An objective for thresholding is to reduce the complexity of the raw data to a binary level or combinations of binary levels. Another reason for thresholding is to obtain an object's silhouette for other processes such as feature extraction.

In some cases, thresholding may not segment the object adequately because the object along with portions of the background are extracted within the range of the threshold. One method to extract distinct **blobs** or regions from such data is by analysis of neighbourhood pixels or **connectivity analysis** [Im, 1992; Cunningham, 1981]. Depending on the application, one could be more interested in the outlines of the object rather than in its region properties. In this case, there are numerous methods to perform edge segmentation [Sonka et al., 1993; Davies, 1990; Haralick and Shapiro, 1992; Gonzalez and Woods, 1992; Fu and Mui, 1981]. Probably the more popular conventional varieties of these are the Roberts, Prewit and Sobel edge detectors. These are also known as gradient operators since they detect intensity gradients. Other exotic methods for edge detection include the morphological operations in [Yang and Li, 1995, Krishnapuram and Gupta, 1992], fuzzy and neural methods in [Pal and King, 1983, Bezdek and Kerr, 1994] or smart heuristics [Cohen, 1993]. In Chapter 7, we illustrate the use of some of these image preprocessing procedures, namely thresholding, connectivity analysis and edge segmentation for general pattern recognition.

The above review is intentionally brief because the landscape of image preprocessing is huge and we do not wish to digress too far from our particular field of interest. Having successfully segmented the object from its environment, the next task is to analyse the data derived from the object for structure. The theories of pattern recognition presented in this chapter provide the basic tools to accomplish this task. In anticipation of this we review some of the basic concepts of cluster analysis.

2.1.3 Definitions of Cluster

Clustering algorithms such as partitional clustering algorithms are oriented to find structure in data, not to establish rules relating to the separation of the data. Everitt [1974] gives three definitions of a cluster:

- A cluster is a set of entities which are alike; entities from different clusters are not alike.

- A cluster is an aggregation of points in the test space such that the distance metric between any two points in the cluster is less than the distance between any two points not in it.
- Clusters may be described as connected regions of a multi-dimensional space containing a relatively high density of points, separated from other such regions containing a relatively low density of points.

While it is undoubtedly easy to give a *functional* definition of a cluster, a *general* definition of a cluster is extremely difficult to define precisely. This is partly because its interpretation depends on subjective processes involving aesthetics, imagination and experiences.

2.1.4 Data Presentation and Types

Raw data for cluster analysis can be presented in two standard formats: (i) as a *pattern* matrix and (ii) as an $n \times d$ *proximity* matrix.

In the pattern matrix format, data is represented by n rows of objects and d columns of measurements or attributes. For example, if students at a university are to be clustered, each object may represent a student and each column, students' responses to a question of a course content. Each row then represents a pattern and each column a feature or measurement. In image analysis, the d features are the orthogonal axes and the n objects are points in the d dimensional space, called a pattern space. The pattern space is the space occupied by these n points. In this patten space, a cluster can be visualised as a collection of points grouped together according to some clustering criteria. A clustering algorithm is particularly useful for identifying the natural groupings in spacings of many dimensions.

The proximity matrix accumulates the indices of proximity or affinity. For example, if the first row of each column and the first column of each row represent the subjects (in same sequential order) prescribed for a course at a university, then the intersections of a row to each column contain the same values as the identical column intersections with each row. The intersection of a row and a column may represent an average of students' subjective response, on a scale of one to ten, of the two subjects' dissimilarity.

The types of data commonly encountered within these two matrix formats are indicated in Fig. 2.1. Pattern matrix data types can be *binary*, *discrete* or *continuous*. The *scale* category indicates the relative significance of numbers that can be either *nominal* or *ordinal* for qualitative data and *interval* or *ratio* for quantitative data. An example of each data type is given below:

- Qualitative nominal: yes = 1, no = 0.
- Qualitative ordinal: 1, 2, 3, 4, 5 where 1 = cold and 5 = hot.
- Quantitative interval: 0 to 100, degrees Celsius (in relative reference scale).
- Quantitative ratio: 0 to 100, degrees Kelvin (in absolute reference scale).

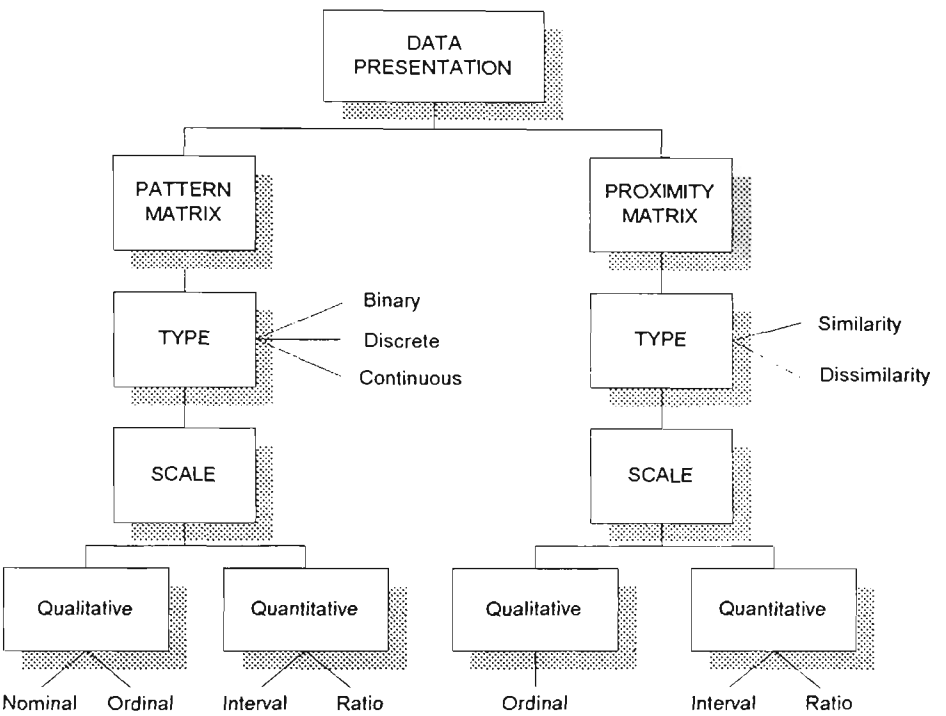


Figure 2.1 Data presentation and types. Adapted from [Jain and Dubes, 1988].

2.1.5 Measures of Proximity

A proximity index (or coefficient) is either a *similarity* or a *dissimilarity* and is defined to reflect maximum value (normally unity) when the condition is true. For example, the Euclidean distance (see Definition A.5, Appendix A) is a dissimilarity index because when the distance is small the index has a low value (or low dissimilarity), and when the distance is large the index has a high value (or high dissimilarity). In contrast, a member-

ship function of a fuzzy clustering algorithm is a similarity index. Denoting two points by i and j, then the proximity indices are related by

$$s(i,j) = 1 - d(i,j) \quad \text{for } 0 \le s(i,j) \le 1 \text{ and } 0 \le d(i,j) \le 1 \tag{2.1.1}$$

where $s(i,j)$ and $d(i,j)$ are the similarity index and dissimilarity index respectively. Denoting a proximity index by $D(i,j)$, then the three properties of a proximity index are:

1 (a). For a dissimilarity, $D(i,i) = 0, \quad \forall i.$

(b). For a similarity, $D(i,i) \ge \max_j D(i,j), \quad \forall i.$

2 $D(i,j) = D(j,i), \quad \forall i,j.$

3 $D(i,j) \ge 0, \quad \forall i,j.$

In fuzzy clustering, a **norm** (or vector length) is used to represent the distance (or metric) between feature vectors or points (see Appendix A for details). If d-component vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ belong in $\Re^d$, then $\|\mathbf{x}_i - \mathbf{x}_j\|$ (or $\|\mathbf{X}\|$ if $\mathbf{X} = \mathbf{x}_i - \mathbf{x}_j$) represents the distance between two vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ (or the length of their vector difference). Alternatively, this distance is denoted by $d_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|$ (also $d(i,j)$ or $d(\mathbf{x}_i,\mathbf{x}_j)$) to represent the distance between two items of data at points i and j. The four unit norms in $\Re^2$ are shown on Fig. 2.2. The distance formula represented by the four norms are summarised in Table 2.1. These norms can be obtained from the Minkowsky metric (except the sup norm)

$$d(i,j) = \left(\sum_{k=1}^{d} |x_{ik} - x_{jk}|^q \right)^{1/q} \quad \text{for } q \ge 1 \tag{2.1.2}$$

The value of q in (2.1.2) corresponds to the norm type (eg. $q = 2$ is a Euclidean norm). The norms of Table 2.1 exert a significant influence over the shape of clusters that are detected in fuzzy clustering. For feature vectors in $\Re^2$, round clusters are obtained from the *Euclidean* norm, elliptic clusters from the *Mahalanobis* norm, and square shape clusters from the *supremum* norm. Higher dimensional norms detect hyperquadrics.

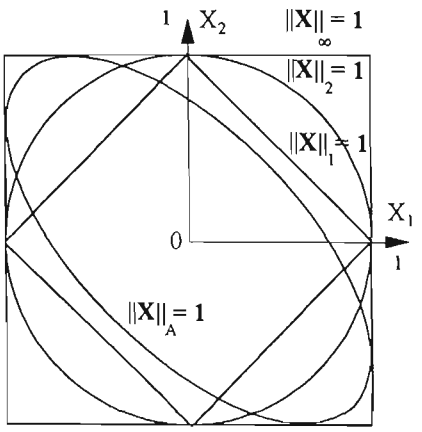


Figure 2.2 Four unit norms of vectors in $\Re^2$. The boundary has a unit norm. Norms less than unity are contained inside the boundary. Adapted from [Bezdek, 1995a].

Name of norm	Distance formula
Mahalanobis	$\|\mathbf{x}_i - \mathbf{v}_j\|^2_{F^{-1}} = (\mathbf{x}_i - \mathbf{v}_j)^T F^{-1} (\mathbf{x}_i - \mathbf{v}_j)$
Euclidean	$\|\mathbf{x}_i - \mathbf{v}_j\|^2_2 = \sum_k (\mathbf{x}_{ik} - \mathbf{v}_{jk})^2$
City block	$\|\mathbf{x}_i - \mathbf{v}_j\|_1 = \sum_k \|\mathbf{x}_{ik} - \mathbf{v}_{jk}\|$
Sup or Max	$\|\mathbf{x}_i - \mathbf{v}_j\|_\infty = \max_k \{\|\mathbf{x}_{ik} - \mathbf{v}_{jk}\|\}$

Table 2.1 Distance formula of four norms used in fuzzy clustering

F^{-1} is a covariance matrix, $\mathbf{v}_j$ denotes the jth cluster prototype and $\mathbf{x}_i$ is the ith feature vector of data set X. Both vectors are in $\Re^d$.

Additional to the norms described above, the similarity measures in Table 2.2 [Diday and Simon, 1976] are useful in conventional cluster analysis.

Name	Similarity measure
Camberra	$d(i,j) = \sum_k \frac{\|x_{ik} - x_{jk}\|}{\|x_{ik} + x_{jk}\|}$
Chi-square	$d(i,j) = \sum_k \frac{1}{\sum_i x_{ik}} \left[\frac{x_{ik}}{\sum_k x_{ik}} - \frac{x_{jk}}{\sum_k x_{jk}} \right]$
Correlation	$d(i,j) = \frac{\sum_k (x_{ik} - \bar{x}_k)(x_{jk} - \bar{x}_k)}{\left[\sum_k (x_{ik} - \bar{x}_k)^2 \sum_k (x_{jk} - \bar{x}_k)^2 \right]^{1/2}}$

Table 2.2 Four similarity measures used in conventional cluster analysis.

To obtain meaningful results in clustering, a meaningful norm has to be established a priori. Also, the choice of variables must contain relevant information. In general, the selection of meaningful variables is a non-trivial task that may involve some common sense, subject matter knowledge and experimentation.

2.1.6 Types of Classifications

The types of classification categories are depicted in Fig. 2.3.

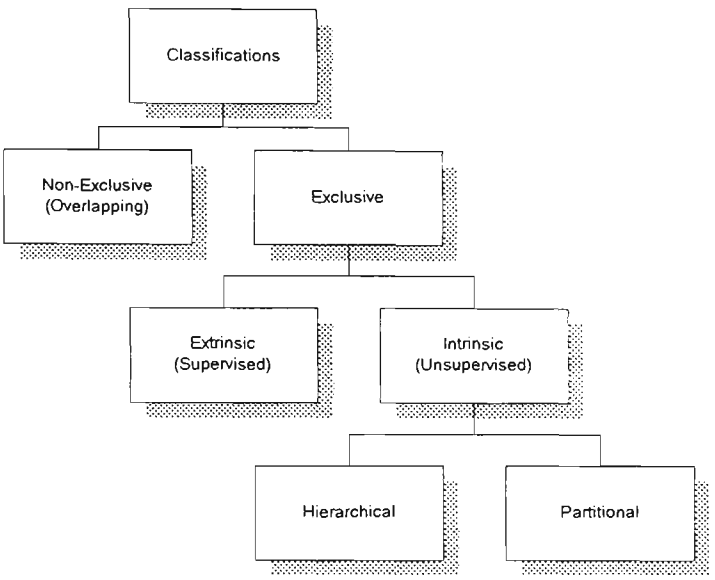


Figure 2.3 Tree of classification types. Adapted from Jain and Dubes [1988].

Exclusive classification refers to subsets of disjoint classes where each object belongs to only one class. Non-exclusive classification allows overlapping of classes such that an object can belong to more than one class. Fuzzy clustering is an example of such a classification. Intrinsic classification is called *unsupervised learning* in pattern recognition because the objects used are unlabelled. Extrinsic classification uses category labels as well as the proximity index on objects. Extrinsic classifiers *learn* to classify from labelled data. This mode of classification is described as *supervised clustering*. An example of this is the Bayes classifier where the number of classes is assumed known. Intrinsic **classifiers** use unlabelled data. Cluster analysis is intrinsic classification. Unsupervised clustering may be divided into two types: (i) hierarchical and (ii) partitional. Partitional clustering is discussed in Section 2.3. Other clustering methods are outside the scope of this thesis. For a discussion of these, refer to [Jain and Dubes, 1988; Kaufman and Rousseeuw, 1990; Everitt, 1974; Gordon, 1981; Har-even and Brailovsky, 1995; Postaire et al., 1993].

Remarks

Cluster analysis and classification are pattern recognition terms with different meanings but sometimes used as if they are alike. In classification, the objective is to find the optimal decision boundaries separating the objects, whereas cluster analysis is concerned with identifying objects (a labelling problem) in the partition space. Classifiers are usually taught the correct groupings from labelled data but clustering algorithms attempt to produce the correct groupings from data that are unlabelled. Roughly speaking, both methods help to identify objects for pattern recognition, although in different ways.

Some of the common methods of pattern recognition includes the following: knowledge based methods [Adimari et al., 1988; Jolion, 1994], rule based methods [Krishnapuram and Keller, 1993b; Rhee and Krishnapuram, 1994], mathematical morphology [Serra, 1988; Dougherty, 1992], Hough transform [Davies, 1990; Leavers, 1992], semantic networks [Nieman et al., 1990] and geometric constraints [Grimson, 1990]. These methods of pattern recognition are outside the scope of the thesis.

2.2 Bayes Decision Theory

Historically, the Bayes decision theory is the basis for most pattern recognition techniques. The objective function of FCM and the clustering criterion of KNN incorporate the Bayes decision rule implicitly. According to Diday and Simon [1976], the field of pattern recognition has developed mostly as statistical classification techniques. It is widely acknowledged that the **Bayes decision rule** gives the optimum classifier performance in the sense of minimum probability of error. This chapter presents the principles of classification for pattern recognition and introduces the discriminant function and its relation to the decision boundary of classes. These concepts are assumed in the fuzzy cluster models.

2.2.1 Bayes Decision Rule – Discrete Case

If there are only two possible states of nature w_1 and w_2, and these occur randomly, then the simplest decision rule for classification is to decide w_1 if the a priori probability $P(w_1)$ is greater than $P(w_2)$. This rule maximises the probability of a correct decision or minimises the probability of error. In general, a decision rule is not limited to a priori probabilities. Suppose a continuous random variable x can be observed such that $p(x|w_j)$ is its conditional probability density function given that the class is w_j. In this case, the Bayes decision rule relates the conditional a posteriori probability $P(w_j|x)$ of a random variable x to its a priori probability $P(w_j)$ according to the relation

$$P(w_j|x) = \frac{p(x|w_j)P(w_j)}{p(x)} \tag{2.2.1}$$

$$p(x) = \sum_{j=1}^{c} p(x|w_j)P(w_j) \tag{2.2.2}$$

where c is the number of classes. Note that the lower case p denotes probability density function and the upper case P the probability distribution function. For an observation x, if $P(w_1|x) > P(w_2|x)$, one would decide class w_1. To justify this procedure, consider the probability of error associated with a particular observation of x:

$$P(error|x) = \begin{cases} P(w_1|x) & \text{if decide } w_2 \\ P(w_2|x) & \text{if decide } w_1 \end{cases}$$

For the same observation x, the probability of error can be minimised by deciding w_1 if $P(w_1|x) > P(w_2|x)$, and w_2 if $P(w_2|x) > P(w_1|x)$. To confirm that this rule minimises the average probability of error, it may be noted that the average probability of error is

$$P(error) = \int_{-\infty}^{\infty} P(error, x)dx = \int_{-\infty}^{\infty} P(error|x)p(x)dx$$

If for every x, $P(error|x)$ is made as small as possible then the integral must be as small as possible. This justifies the *Bayes decision rule* for minimising the probability of error given by:

Bayes decision rule.

- decide the class w_1 if $P(w_1|x) > P(w_2|x)$ and
- decide the class w_2 if $P(w_2|x) > P(w_1|x)$.

This rule emphasises the role of the a posteriori probabilities. The Bayes decision rule in terms of the conditional and a priori probabilities (ignoring the scale factor of (2.2.2)) is:

- decide the class w_1 if $p(x|w_1)P(w_1) > p(x|w_2)P(w_2)$ and
- decide the class w_2 if $p(x|w_2)P(w_2) > p(x|w_1)P(w_1)$.

2.2.2 Bayes Decision Rule – Continuous Case

The Bayes decision rule can be generalised for the continuous case with the following conditions:

- more than two class states
- more than one feature
- a loss function

Let the loss function for all errors be equally costly. Let $\Omega = \{w_1,...,w_c\}$ be the c finite number of classes and $A = \{\alpha_1,...,\alpha_a\}$ be the a possible actions. Let $\lambda(\alpha_i|w_j)$ be the loss incurred for action α_i when the true class is w_j. Let feature vector $\mathbf{x}$ be d component and $p(\mathbf{x}|w_j)$ be the probability density function for $\mathbf{x}$ conditioned on class w_j. Then the Bayes rule is

$$P(w_j|\mathbf{x}) = \frac{p(\mathbf{x}|w_j)P(w_j)}{p(\mathbf{x})} \qquad (2.2.3)$$

$$p(\mathbf{x}) = \sum_{j=1}^{c} p(\mathbf{x}|w_j)P(w_j) \tag{2.2.4}$$

Given that the loss associated with α_i is $\lambda(\alpha_i|w_j)$ when the true state is w_j, and $P(w_j|\mathbf{x})$ is the probability that the state is w_j, then the risk associated with α_i is

$$R(\alpha_i|\mathbf{x}) = \sum_{j=1}^{c} \lambda(\alpha_i|w_j)P(w_j|\mathbf{x}) \tag{2.2.5}$$

$R(\alpha_i|\mathbf{x})$ is known as the *conditional risk*. For an observation of $\mathbf{x}$, the expected loss is minimised by selecting the action that minimises the conditional risk. Therefore the Bayes rule that minimises the overall risk is to compute the conditional risk given by (2.2.5) for every possible action α_i, for $i = 1,...,a$ and to select the α_i for which $R(\alpha_i|\mathbf{x})$ is a minimum. The resulting overall risk is called the *Bayes risk* and is the best that can be achieved.

2.2.3 Two Category Classification

Denoting $\lambda_{ij} = \lambda(\alpha_i|w_j)$ as the loss incurred for deciding w_i when the true class is w_j, then (2.2.5) can be reformulated for a two class case as

$$R(\alpha_1|\mathbf{x}) = \lambda_{11}P(w_1|\mathbf{x}) + \lambda_{12}P(w_2|\mathbf{x}) \tag{2.2.6}$$

$$R(\alpha_2|\mathbf{x}) = \lambda_{21}P(w_1|\mathbf{x}) + \lambda_{22}P(w_2|\mathbf{x}) \tag{2.2.7}$$

The Bayes rule that minimises the conditional risk is to decide w_1 if $R(\alpha_1|\mathbf{x}) < R(\alpha_2|\mathbf{x})$. The corresponding a posteriori probabilities of the Bayes rule is to decide w_1 if

$$(\lambda_{21} - \lambda_{11})P(w_1|\mathbf{x}) > (\lambda_{12} - \lambda_{22})P(w_2|\mathbf{x}) \tag{2.2.8}$$

The Bayes rule in terms of the a priori probabilities is to decide w_1 if

$$(\lambda_{21} - \lambda_{11})p(\mathbf{x}|w_1)P(w_1) > (\lambda_{12} - \lambda_{22})p(\mathbf{x}|w_2)P(w_2) \tag{2.2.9}$$

2.2.4 Minimum Error-Rate Classification

A rule that minimises the average probability of error or error rate is given by the loss function

$$\lambda(\alpha_i|w_j) = \begin{cases} 0 & i = j \\ 1 & i \neq j \end{cases} \quad \text{for } i,j = 1,...,c \tag{2.2.10}$$

This equation is also known as the *zero-one* loss function. If a decision is correct ($i = j$) it assigns no loss but if a decision is incorrect ($i \neq j$) it assigns a loss of one unit. Thus all errors are equally costly. Substituting (2.2.10) into (2.2.5) gives the conditional risk

$$R(\alpha_i|\mathbf{x}) = \sum_{j=1, i\neq j}^{c} P(w_j|\mathbf{x}) = 1 - P(w_i|\mathbf{x}) \tag{2.2.11}$$

where $P(w_i|\mathbf{x})$ is the conditional probability that α_i is the correct action. Thus the Bayes decision rule to minimise the conditional risk is equivalent to the rule that minimises the average probability of error (or error rate) given by a posteriori probabilities. Decide w_i if

$$P(w_i|\mathbf{x}) > P(w_j|\mathbf{x}) \qquad \text{for all } i \neq j \tag{2.2.12}$$

or in terms of the conditional probability density function for $\mathbf{x}$

$$p(\mathbf{x}|w_i)P(w_i) > p(\mathbf{x}|w_j)P(w_j) \text{ for all } i \neq j \tag{2.2.13}$$

2.2.5 Classifiers, Discriminant Functions and Decision Surfaces

In representing pattern classifiers, it is convenient to define **discriminant** (or decision) **functions** $d_i(\mathbf{x})$, $i = 1,...,c$ such that $d_i(x)$ is maximum for class w_i and

$$d_i(\mathbf{x}) > d_j(\mathbf{x}) \text{ for all } i \neq j \tag{2.2.14}$$

For minimum average error, the a posteriori probability is the maximum discriminant function expressed by

$$d_i(\mathbf{x}) = P(w_i|\mathbf{x}) \tag{2.2.15}$$

Since the choice of $d_i(\mathbf{x})$ is not unique, it can also be represented as

$$d_i(\mathbf{x}) = p(\mathbf{x}|w_i)P(w_i) \tag{2.2.16}$$

or by taking the natural logarithm

$$d_i(\mathbf{x}) = \ln p(\mathbf{x}|w_i) + \ln P(w_i) \tag{2.2.17}$$

Note that the discriminant functions of (2.2.15), (2.2.16) and (2.2.17) are obtained from minimum error-rate classification. The discriminant function divides the feature space comprising of regions of classes according to the rule:

assign to class w_i if $d_i(\mathbf{x}) > d_j(\mathbf{x})$ where $i \neq j$.

More commonly, this is expressed as

$$d(\mathbf{x}) = d_i(\mathbf{x}) - d_j(\mathbf{x})$$

Ties where $d(\mathbf{x}) = 0$ are resolved arbitrarily. The discriminant function of (2.2.17) for a two class case is

$$d(\mathbf{x}) = \ln\frac{P(\mathbf{x}|w_1)}{P(\mathbf{x}|w_2)} + \ln\frac{P(w_1)}{P(w_2)} \tag{2.2.18}$$

2.2.6 Discriminant Functions For Normal Density

A general solution for (2.2.17) can be obtained by approximating the densities $p(\mathbf{x}|w_i)$ with a multivariate normal (Gaussian) density function given by

$$N(\boldsymbol{\mu},\Sigma)=\frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}}\exp\left[-\tfrac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right] \tag{2.2.19}$$

where

$\mathbf{x}=[x_1,x_2,...,x_d]^T$ is a d-component random variable vector

$\boldsymbol{\mu}=[\mu_1,\mu_2,...,\mu_d]^T$ is a d-component mean vector

$\Sigma = \mathrm{E}[(\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^T] = \begin{bmatrix} \sigma_{11} & \sigma_{12} & .. & \sigma_{1d} \\ \sigma_{21} & .. & .. & .. \\ .. & .. & .. & .. \\ \sigma_{d1} & .. & .. & \sigma_{dd} \end{bmatrix}$ is a $d \times d$ symmetric positive semi-definite covariance matrix.

$\sigma_{ik} = \mathrm{E}(x_i-\mu_i)(x_k-\mu_k) = \sigma_{ki}$.

$\boldsymbol{\mu} = \mathrm{E}[\mathbf{x}]$, the expectation of $\mathbf{x}$.

$(\mathbf{x}-\boldsymbol{\mu})^T$ is the transpose of $(\mathbf{x}-\boldsymbol{\mu})$.

Σ^{-1} is the inverse of Σ.

$|\Sigma|$ is the determinant of Σ.

The contours of constant density are the hyperellipsoids given by

$$r^2 = (\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}) \tag{2.2.20}$$

where the principal axes are the eigenvectors and the lengths of these axes are determined by their eigenvalues. Equation (2.2.20) is also known as the squared Mahalanobis distance. The contours of constant density are the hyperellipsoids of constant Mahalanobis distance from $\mathbf{x}$ to $\boldsymbol{\mu}$. The volume of the hyperellipsoid corresponding to the Mahalanobis distance r is

$$V = V_d|\Sigma|^{1/2}r^d \tag{2.2.21}$$

where

$$V_d = \begin{cases} \dfrac{\pi^{d/2}}{(\frac{d}{2})!} & d \quad even \\ \dfrac{2^d\pi^{(d-1)/2}\left(\frac{d-1}{2}\right)!}{d!} & d \quad odd \end{cases} \tag{2.2.22}$$

For a given dimensionality d, the scatter of the samples varies directly with $|\Sigma|^{1/2}$. The discriminant function of (2.2.17) can be solved by approximating $p(\mathbf{x}|w_i)$ with (2.2.19) to give

$$d_i(\mathbf{x}) = -\tfrac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T \Sigma_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i) - \tfrac{d}{2}\ln 2\pi - \tfrac{1}{2}\ln|\Sigma_i| + \ln P(w_i) \qquad (2.2.23)$$

This equation has some interesting features which are discussed under three cases.

Case 1: $\Sigma_i = \sigma_{kk}I$

In this case, the covariance matrix is diagonal with σ_{kk} times the identity matrix I. The features are statistically independent and each feature has the same variance σ_{kk} (defined under Eq. (2.2.19) for $k = 1,\ldots,c$). Geometrically, this corresponds to samples of equal size hyperspherical clusters in which the ith cluster is centred at mean vector $\boldsymbol{\mu}_i$. Since both $\frac{1}{2}\ln|\Sigma_i|$ and $\frac{d}{2}\ln 2\pi$ are constant terms, these may be ignored to give

$$d_i(\mathbf{x}) = -\frac{\|\mathbf{x}-\boldsymbol{\mu}_i\|^2}{2\sigma_{kk}} + \ln P(w_i) \qquad (2.2.24)$$

where

$\|\mathbf{x} - \boldsymbol{\mu}_i\|^2 = (\mathbf{x} - \boldsymbol{\mu}_i)^T(\mathbf{x} - \boldsymbol{\mu}_i)$ is the Euclidean norm (see Appendix A).

If the a priori probabilities $P(w_i)$ are the same for all c classes then the $\ln P(w_i)$ terms can also be ignored. In this case, the optimum classification rule is to assign $\mathbf{x}$ to the class of the nearest mean. Such a classifier is called the minimum distance classifier. The optimum classification rule (or least squares error) is the basis of numerous clustering algorithms. It is the clustering criterion of KNN and the objective function in the FCM, weighted by the fuzzy memberships. If the $\ln P(w_i)$ terms are not equal, then (2.2.24) can be expressed as

$$d_i(\mathbf{x}) = -\frac{1}{2\sigma_{kk}}[\mathbf{x}^T\mathbf{x} - 2\boldsymbol{\mu}_i^T\mathbf{x} + \boldsymbol{\mu}_i^T\boldsymbol{\mu}_i] + \ln P(w_i) \qquad (2.2.25)$$

Since $(\mathbf{x}^T\mathbf{x})$ is the same for all i, it may be ignored so that (2.2.25) yields the linear discriminant functions

$$d_i(\mathbf{x}) = \mathbf{g}_i^T\mathbf{x} + h \qquad (2.2.26)$$

where

$$\mathbf{g}_i = \frac{1}{\sigma_{kk}}\boldsymbol{\mu}_i \qquad (2.2.27)$$

$$h = -\frac{1}{2\sigma_{kk}}\boldsymbol{\mu}_i^T\boldsymbol{\mu}_i + \ln P(w_i) \qquad (2.2.28)$$

At this point, note that **hyperplanes** are defined by the linear discriminant functions

$$d_i(\mathbf{x}) = d_j(\mathbf{x}) \tag{2.2.29}$$

Substituting (2.2.27) and (2.2.28) into (2.2.29) and noting the different indices, a decision plane for two classes w_i and w_j with the same variance σ_{kk} is defined by

$$d_i(\mathbf{x}) - d_j(\mathbf{x}) = \frac{1}{\sigma_{kk}}(\mu_i^T - \mu_j^T)\mathbf{x} - \frac{1}{2\sigma_{kk}}(\mu_i^T \mu_i - \mu_j^T \mu_j) + M \tag{2.2.30}$$

where $M = \ln \dfrac{P(w_i)}{P(w_j)}$

From (2.2.30) the hyperplane through the point $\mathbf{x}_0$ is given by

$$\mathbf{x}_0 = \frac{\dfrac{1}{2\sigma_{kk}}(\mu_i^T \mu_i - \mu_j^T \mu_j) - M}{\dfrac{1}{\sigma_{kk}}(\mu_i^T - \mu_j^T)}$$

which simplifies to

$$\mathbf{x}_0 = \frac{1}{2}(\mu_i + \mu_j) - \frac{\sigma_{kk}}{\|\mu_i - \mu_j\|^2} \ln \frac{P(w_i)}{P(w_j)}(\mu_i - \mu_j) \tag{2.2.31}$$

This hyperplane plane defined by (2.2.31) represents the decision boundary that separates the two regions of the classes w_i and w_j and has the following properties:

- It is orthogonal to the vector $(\mu_i - \mu_j)$. The right side second term of (2.2.31) is the projection of $(\mu_i + \mu_j)$ onto $(\mu_i - \mu_j)$. Thus the vector difference of the two terms of (2.2.31) is the hyperplane vector through $\mathbf{x}_0$, orthogonal to the vector $(\mu_i - \mu_j)$.
- If $P(w_i) = P(w_j)$ then $\mathbf{x}_0 = (\mu_i + \mu_j)/2$, and the hyperplane plane bisects the means.
- If $P(w_i) \neq P(w_j)$ then the point $\mathbf{x}_0$ shifts away from the more likely mean.
- If $\sigma_{kk} << (\mu_i - \mu_j)$ then the position of the decision boundary is insensitive to the values of the a priori probabilities.

The above result has practical implications for neural networks functioning as classifiers. It has been demonstrated theoretically by [Ruck et al., 1990] that the backpropagation neural network (introduced in Section 2.5) approximates to the optimum decision boundary of (2.2.31). This result also suggests a clustering criterion for clustering algorithm. For the image data, the probabilities of $\mathbf{x}$ are normally uncorrelated and are equally likely. Therefore the data approximates Case 1 with $P(w_i) = P(w_j)$. The optimum cluster boundary (in the Bayes sense) is then defined by the mean of the two closest cluster **prototypes**. This rule is realised in practice by assigning a point x to its nearest cluster according to the criterion: decide $x \in w_i$ if

$$\min |x - v_i|^2 < |x - v_j|^2, \ \forall i \neq j$$

where v_i are the cluster centres corresponding to classes w_i. Section (2.3.3) shows how this rule is implemented in the KNN algorithm.

Case 2: $\Sigma_i = \Sigma$ (defined by (2.2.19))

For this case, the covariance matrices for all c classes are identical and the samples fall into hyperellipsoidal clusters of equal size and shape. Since both $|\Sigma_i|$ and $\frac{d}{2}\ln 2\pi$ are independent of i in (2.2.23) they can be ignored as additive constants. The discriminant functions are given by

$$d_i(\mathbf{x}) = -\tfrac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_i)^T \Sigma_i^{-1}(\mathbf{x} - \boldsymbol{\mu}_i) + \ln P(w_i) \tag{2.2.32}$$

Assuming equal probabilities for all c classes, $\ln P(w_i)$ can also be ignored yielding

$$d_i(\mathbf{x}) = -\tfrac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_i)^T \Sigma_i^{-1}(\mathbf{x} - \boldsymbol{\mu}_i) \tag{2.2.33}$$

which implies that the optimum decision boundary is just the squared Mahalanobis distance of $\mathbf{x}$ to each of the c mean vectors $\boldsymbol{\mu}_i$. The bivariate normal density function of (2.2.33) has some interesting geometric properties that also appear in fuzzy clustering (see Section 2.4). This distribution function has the expression

$$(\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu}) = \frac{1}{1 - \rho_{12}^2}\left[\left(\frac{x_1 - \mu_1}{\sqrt{\sigma_{11}}}\right)^2 + \left(\frac{x_2 - \mu_2}{\sqrt{\sigma_{22}}}\right)^2 - 2\rho_{12}\left(\frac{x_1 - \mu_1}{\sqrt{\sigma_{11}}}\right)\left(\frac{x_2 - \mu_2}{\sqrt{\sigma_{22}}}\right)\right] \tag{2.2.34}$$

where

$$\mathbf{x} = [x_1, x_2]^T$$

$$\boldsymbol{\mu} = [\mu_1, \mu_2]^T$$

$$\Sigma^{-1} = \frac{1}{\sigma_{11}\sigma_{22} - \sigma_{12}^2}\begin{bmatrix} \sigma_{22} & -\sigma_{12} \\ -\sigma_{12} & \sigma_{11} \end{bmatrix}$$

$$\rho_{12} = \frac{\sigma_{12}}{\sqrt{\sigma_{11}}\sqrt{\sigma_{22}}}$$

Note that if the random variables x_1 and x_2 are uncorrelated, then the joint correlation coefficient $\rho_{12} = 0$ and the joint density can be expressed as the product of two univariate normal densities similar to the form of (2.2.19). For the two class case, the following constant density contours are obtained from (2.2.34):

(a) Circular contour: If $\sigma_{11} = \sigma_{12} = \sigma > 0$ and $\rho_{12} = 0$.

(b) Elliptic contour: If $\sigma_{11} = \sigma_{12} = \sigma > 0$ and $0 < \rho_{12} < 1$.

(c) Linear contour: If $\sigma_{11} = \sigma_{12} = \sigma > 0$ and $\rho_{12} = 1$.

Following the procedure of Case 1, the result for the multivariate case is given by

$$\mathbf{x}_0 = \frac{1}{2}(\mu_i + \mu_j) - \frac{\ln \dfrac{P(w_i)}{P(w_j)}}{(\mu_i - \mu_j)^T \Sigma^{-1} (\mu_i - \mu_j)} (\mu_i - \mu_j) \tag{2.2.35}$$

The decision boundary of (2.2.35) is a hyperplane like Case 1. However, the differences of vectors ($\mu_i - \mu_j$) are generally not orthogonal to the hyperplanes, although the planes do bisect the lines between the means.

Case 3: Σ_i Arbitrary

For the multivariate case, the $\frac{d}{2}\ln 2\pi$ term can be ignored resulting in

$$d_i(\mathbf{x}) = \mathbf{x}^T W_i \mathbf{x} + \mathbf{w}_i^T \mathbf{x} + h \tag{2.2.36}$$

$$W_i = -\tfrac{1}{2}\Sigma_i^{-1} \tag{2.2.37}$$

$$\mathbf{w}_i = \Sigma_i^{-1} \mu_i \tag{2.2.38}$$

$$h = -\tfrac{1}{2}\mu_i^T \Sigma_i^{-1} \mu_i - \tfrac{1}{2}\ln|\Sigma_i| + \ln P(w_i) \tag{2.2.39}$$

The discriminant function of (2.2.36) is quadratic in $\mathbf{x}$ and the decision surfaces are hyper-quadrics that can assume any of the forms of pairs of hyperplanes, hyperspheres, hyper-ellipsoids, hyperparaboloids and hyperhyperboloids of various types.

Remarks

In classification problems, the Bayes rule for minimising the probability of error is to choose class w_i which maximises the a posteriori probability $P(w_i|\mathbf{x})$. This probability can be calculated from the a priori probabilities $P(w_i)$ and conditional densities $p(\mathbf{x}|w_i)$. The main problem in using Bayes rule for pattern recognition is that the conditional densities $p(\mathbf{x}|w_i)$ are rarely known. If $p(\mathbf{x}|w_i)$ can be approximated by $N(\mu_i, \Sigma_i)$, then a discriminant function given by (2.2.23) can be used for optimal classification. Although widely used, the Bayes rule is not the only statistical criterion for classification. James [1985] gives other criteria of classification. Further details on statistical classification can be found in [Jain, 1987; Duda and Hart, 1973; Fukunaga, 1972].

2.3 Partitional Clustering Theory

Section 2.2 showed that an optimum discriminant function and thus the optimum partition, can be obtained from the Bayes decision rule if the class-conditional densities are Gaussian. If the forms of the class-conditional densities are unknown, one can use non-parametric procedures of the partitional clustering type like the k-Nearest Neighbour (KNN) algorithm. The chapter begins with features of partitional clustering in Sections 2.3.1 to 2.3.2, followed by a discussion of the KNN algorithm in Section 2.3.3 and cluster validity in Section 2.3.4.

2.3.1 Definition and Properties

Let E be a set of N elements (or objects) $\mathbf{x}_i$ in $\Re^d$. A partition of $P = \{P_1, P_2, \ldots, P_s, \ldots, P_c\}$ is a set of subsets of E such that $P_i \cap P_j = \varnothing$, for all i, j and $\bigcup_{j=1,\ldots,c} P_j = E$. Partitional clustering obtains a partition of a set E of N objects $\mathbf{x}_i$ by the use of one or more proximity indices. A partition P is a set of disjoint subsets of E. An element P_s of P is a cluster. The goal of clustering is to define a mapping T such that

$$T : \mathbf{x}_i \rightarrow P_s \text{ for } i = 1, \ldots, N.$$

2.3.1.1 Homogeneity

A cluster P_s is said to be homogeneous

if and only if $\mathbf{x}_i, \mathbf{x}_j \in P_s$ and $\mathbf{x}_k \notin P_s$, or

$d(\mathbf{x}_i, \mathbf{x}_j) \leq d(\mathbf{x}_i, \mathbf{x}_k)$ and $d(\mathbf{x}_i, \mathbf{x}_j) \leq d(\mathbf{x}_j, \mathbf{x}_k)$

for $i, j, k = 1, \ldots, N$. A partition P is said to be homogeneous if the above property is true for all $P_s \in P$. This statement means that two elements of the same cluster are more similar than to any other cluster of E. Similarity is defined via a proximity index (Section 2.1.5).

2.3.1.2 Stability of a Partition

A good partition minimises a criterion function J. For example, let

$$\text{Sup } P_s = \max [d(\mathbf{x}_i, \mathbf{x}_j)] \text{ for } \mathbf{x}_i, \mathbf{x}_j \in P_s.$$

Let P_s' be a cluster obtained from P_s by removing some objects and taking some new ones but having the same total number of objects. If Sup P_s > Sup P_s', P_s is said to be better than P_s' in the sense of the criterion. If no other P_s' better than P_s can be found, P_s is said to be stable.

2.3.2 Criterion Function

Partitional methods generally expect data as a pattern matrix and assume the data is measured on a ratio scale. If the features are on a nominal or ordinal scale, the data types do not give meaningful cluster centres based on Euclidean distances. In this case, hierarchical methods of clustering are used.

In Section 1.1.2, a problem with data dimensions was noted. One way of avoiding a combinatorial explosion is to evaluate a criterion function on a small set of reasonable partitions. This solution involves optimising a criterion function using gradient ascent or iterative hill climbing techniques. Starting from an initial partition, objects are moved from one cluster to another to improve the criterion function. Each successive partition represents a perturbation of the previous partition. Algorithms based on this technique can be computation efficient but may or may not converge to a global minimum of the criterion function. Other approaches to solve this problem are described in [Jensen,1969; Rao, 1971; Koontz et al., 1975; Lefkovich, 1980].

Since clusters can assume any arbitrary shape and size, it is difficult to conceive a criterion function that is equally good for all clusters. Many criterion functions have been proposed in the literature, some appearing in different guises. Examples of these are the mutual near-neighbour clustering of Gowda and Krishna [1978], the Maximal Spanning Trees (MST) of Zahn [1971], shared near neighbour MST in Jarvis [1978], mode seeking partitional clustering in Kittler [1976] and the bootstrap approach of Moreau [1987]. Possibly the most common cluster criterion is the Least Squared Error criterion (LSE). This criterion partitions the data for fixed number of clusters that minimise the squares of the errors.

The procedure for LSE is as follows. For a data set of N points in $\Re^d$, let the c cluster set $\{c_1, c_2,...,c_c\}$ have corresponding set of points $\{n_1, n_2,...,n_c\}$, and each object belongs only to its cluster so that

$$\sum_k n_k = N \tag{2.3.1}$$

The mean of the kth cluster is the centroid defined by

$$\mu_k = \frac{1}{n_k}\sum_{i=1}^{n_k} \mathbf{x}_{ki} \tag{2.3.2}$$

where $\mathbf{x}_{ki}$ is the ith element of the data set belonging to the kth cluster c_k. If a Euclidean metric is used, then the least squares error for cluster c_k is

$$e_k^2 = \sum_{k=1}^{n_k} (\mathbf{x}_{ki} - \mu_k)^T (\mathbf{x}_{ki} - \mu_k) \tag{2.3.3}$$

In general, the LSE can be weighted by any $d \times d$ symmetric positive definite matrix A so that

$$e_{k,A}^2 = \sum_{k=1}^{n_k} (\mathbf{x}_{ki} - \mu_k)^T A(\mathbf{x}_{ki} - \mu_k) \tag{2.3.4}$$

In particular, for a Euclidean distance, $A = I$ ($d \times d$ identity matrix), and for a Mahalanobis distance, $A = \Sigma^1$ is a covariance matrix. The geometric forms of these constant norm loci are hyperellipsoids centred at the means μ_k described in Section 2.2. The LSE criterion function finds partitions of fixed c clusters that minimises the sum of the within cluster variations

$$E_k^2 = \sum_{k=1}^{c} e_{k,A}^2 \tag{2.3.5}$$

An optimal partition is defined as one which minimises E_k. Clusters of this function are called *minimum variance* partitions. The kinds of data that are suited to the LSE criterion function are those which form compact clusters that are well separated from one another. However, the LSE criterion is sensitive to outliers because the splitting of clusters, from the influence of outliers, are favoured over one that maintains the integrity of clusters.

2.3.3 *k*-Nearest Neighbour

The KNN is a non-parametric method in which k stands for the number of data partitions (generated from initial k neighbour points). Several variants of KNN exist, essentially to improve the control of cluster development [Anderberg, 1973; Dubes and Jain, 1980]. For the purpose of comparison with our algorithms of Chapter 3, we will only describe the procedure for the basic KNN algorithm.

A basic KNN algorithm.

Step 1. Select k elements of $\mathbf{x}_i$ as prototypes μ_i of the k clusters $\{c_1, c_2, \ldots, c_k\}$.

Step 2. For all remaining data, assign $\mathbf{x}_i$ to cluster j if $d(\mathbf{x}_i, \mu_j)$ is the minimum for $j=1,\ldots,k$.

Step 3. For all j, compute new mean μ_j.

Step 4. If $\|\mu_{j,t} - \mu_{j,t-1}\| < \delta$, Stop, else go to Step 2.

Note: δ controls the stopping condition and t is an iteration index.

The KNN has four main characteristic features, described as follows:

2.3.3.1 Partition Forming

Partitions are formed by specifying the number of clusters k. The positions of the k cluster centres can be randomly selected or based on some heuristics that spread the centres in the vicinity of the data centroid. The partition is developed by each point attaching itself to the nearest cluster prototype. The centroid of a resulting cluster becomes the cluster prototype for the next iteration of clustering until eventually, all cluster prototype values stop changing. The clustering criterion of KNN is of the LSE type. Consequently, different initial partitions may not yield the same final result especially if the clusters are not well separated. This problem is identified as a convergence to local minima. One way to overcome this problem is to run the algorithm with different initial partitions, but the problems of cluster validity remain largely unresolved.

2.3.3.2 Partition Update

Partitions are updated by computing local cluster centroids (centres) and so reduce the square error. There are two main variants to the update of partitions. The method of McQueen [1967] updates the cluster centre immediately on assignment of each data point. Forgy's method [1965] updates all cluster centres after all data points have been assigned to the clusters. The two most commonly used distance metrics are the Euclidean and the Mahalanobis metrics. The Mahalanobis metric is more complex to use as it requires an invertible symmetric positive definite matrix. However, the Mahalanobis metric is a more general metric compared to the Euclidean metric.

2.3.3.3 Cluster Adjustment

Cluster adjustment refers to the adjustment of cluster numbers after certain conditions are met. This capability allows an algorithm to recover from poor cluster results. For example, a cluster is split if it has too many patterns and a large spread along a feature. Two clusters may merge if their clusters are sufficiently close. In most cases however, only ad hoc rules can be applied because general rules cannot deal with the complexity of cluster structures. The presence of outliers can also significantly distort the cluster structures of KNN (except the fuzzy structures derived from PFCM or the possibilistic varieties).

2.3.3.4 Convergence

Convergence of cluster prototypes or centres, in the context of clustering, can be understood in two ways. In the one case, convergence is analytically determined, such as the case with FCM for which convergence theorems exist. In the other, convergence is coerced by rules or algorithmic procedures such as the KNN and most partitional algorithms. At the point of convergence, cluster prototypes may not correspond to their cluster centroids. This occurs in fuzzy clustering algorithms such as the FCM at normal m values. Partitional algorithms do not have automatic stopping points. The algorithm may converge in the sense that the cluster prototypes can be confined to a small radius for larger number of iterations. The stopping point is usually decided on the basis of negligible change in the prototype value between successive iterations. There is no guarantee that the prototypes will converge to global minima. To ensure convergence, a maximum number of iterations may be specified, although the KNN has a reputation for fast convergence (see article [Juan and Vidal, 1994], for a faster k centroid clustering). Partitional clustering algorithms require the number of partitions to be given or known a priori. Consequently, cluster validity is an important consideration for these type of algorithms.

2.3.4 Cluster Validity

Cluster validity addresses the question of cluster number optimality or how many good clusters there are in the data. As Fig. 2.4 shows, this question is neither trivial nor simple to answer. The plus symbols denote the 32 points of some data in 2D real space. The elliptic boundary shown on Fig. 2.4 is not part of the cluster. The boundary is drawn to show the possible cluster structure as detected by an algorithm such as the Gustafson-Kessel

algorithm [1979]. The four different possible cluster substructures in Fig. 2.4 indicate some practical problems in determining cluster validity. A cluster validity measure or index provides an objective basis to make a correct selection of the cluster partitions. One simple validity measure is to assess a change in the cluster criterion function by running the algorithm for a range of cluster numbers, then selecting the cluster number that gives the best result. This may involve some trial and error.

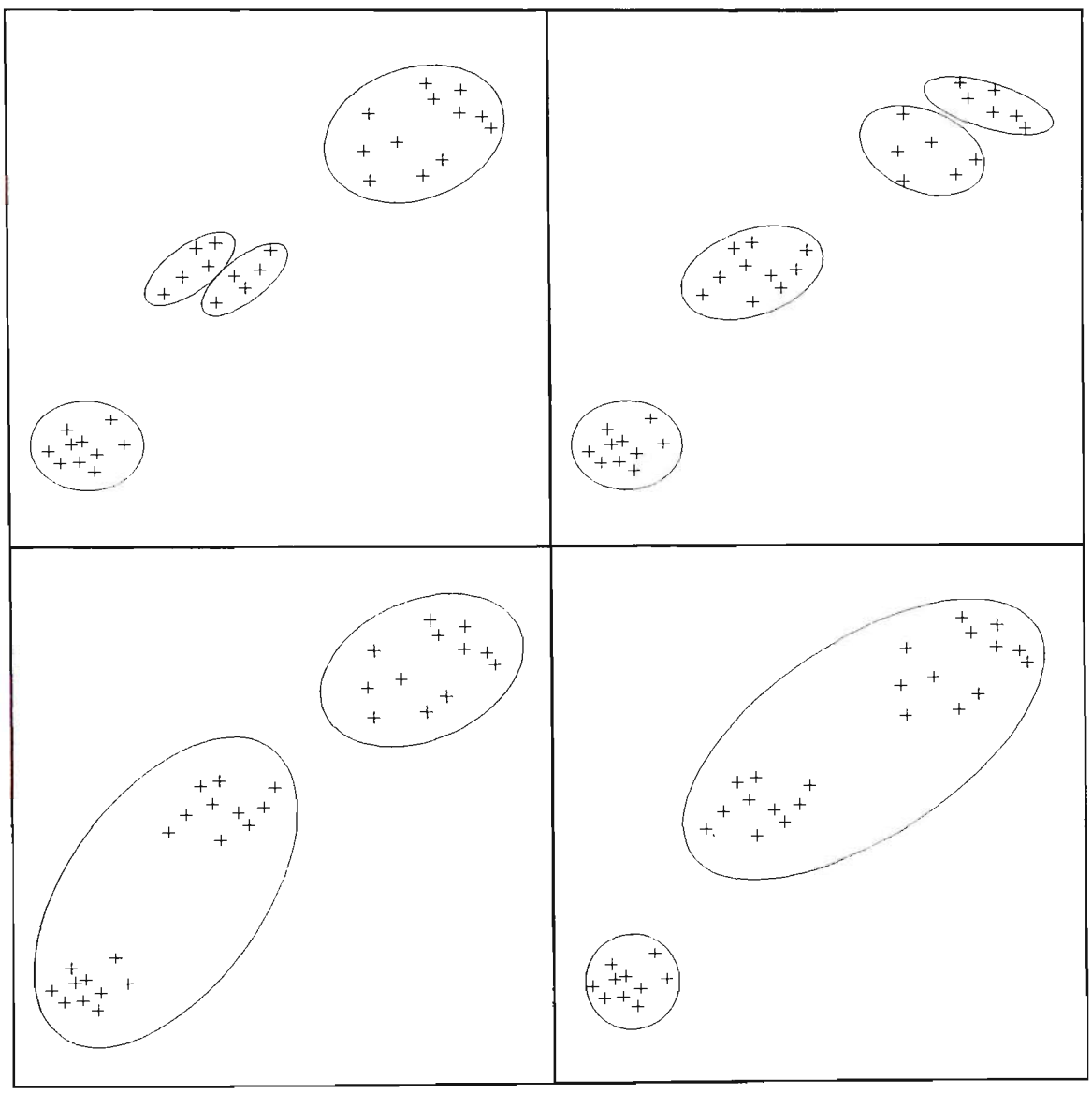


Figure 2.4 Four possible ways of clustering 32 points in $\Re^2$.

A more formal approach is to devise a measure of the goodness of fit to indicate how well a given c cluster description matches the data. The classical measures of fit are the *Chi-square* and the *Kolmogorov-Smirnov* statistics, but the problem of dimensionality precludes their direct application, except for the simplest of cases.

The topic of cluster validity covers a wide field of many different methods. Bezdek [1995b] identifies three different categories of cluster validity: (i) direct methods, (ii) indirect methods and (iii) performance based. The direct methods are the (a) Hubert statistics [Hubert, 1985], (b) Davies-Bouldin index [Davies and Bouldin, 1979], (c) Dunn's Compact and Separated index [Dunn, 1974] and (d) Generalised Dunn's indices [Dunn, 1976]. The indirect methods are the (a) Partition Coefficient [Bezdek, 1974; Zadeh, 1965], (b) Partition Entropy [Shannon, 1948], (c) Xie-Beni and Extended Xie-Beni indices [Xie and Beni, 1991] and (d) Fukuyama-Sugeno index. The performance based method is exemplified by Backer and Jain's fuzzy set decomposition measure [Backer and Jain, 1981].

Other measures of cluster validity are given in [Jain and Dubes, 1988; Kaufman and Rousseeuw, 1990; Bezdek et al., 1980; Dubes, 1987]. Section 3.3 describes our procedure to construct a cluster validity based on the Bayes optimum discriminant function. The following caveats are given to guide the sensible use of cluster validity [Bezdek, 1995b].

1. Numerical representation may not mean adequate powers of discrimination.
2. Algorithms used may not extract structure from data.
3. Appropriate parameters of an algorithm may never be used.
4. The validity indices may give an incorrect cluster interpretation.

Remarks

More details on the partitional clustering algorithms are given in: [Hartigan, 1975; Duda and Hart, 1973; Tou and Gonzalez, 1974; Jain and Dubes, 1988; Kaufman and Rousseeuw, 1990].

2.4 Fuzzy Clustering Theory

Fuzzy sets [Zadeh, 1965] manipulate data and information that possess nonstatistical uncertainty. According to Zadeh, "the fuzzy set was conceived as a result of an attempt to come to grips with the problem of pattern recognition in the context of imprecisely defined categories" [Bezdek, 1981, p. v]. Zadeh coined the audacious term "fuzzy" because it was concrete, immediate and descriptive. The successful applications of fuzzy logic are summarised in [Bezdek, 1995b].

This chapter introduces the theories of fuzzy clustering and examines the analytic procedures for solving fuzzy objective functions. It begins with the first systematic exposition of the fuzzy clustering method of Ruspini in Section 2.4.2, followed by the Duda and Hart's Hard c-Means (HCM) in Section 2.4.3 and Bezdek's Fuzzy c-Means (FCM) in Section 2.4.4. This leads to Bezdek and Im's Possibilistic Fuzzy c-Means (PFCM) algorithm introduced in Section 2.4.5. Clustering properties of PFCM are discussed and compared with FCM. Objective functions with different kinds of membership constraints are presented in Sections 2.4.6 and 2.4.7. These include the variable norms of Gustafson and Kessel's algorithm, and Gath and Geva's algorithm in Section 2.4.6. Next, Krishnapuram and Keller's Possibilistic c-Means (PCM) algorithm, and Bezdek and Im's Enhanced Possibilistic c-Means (EPCM) algorithm are presented in Section 2.4.7. The clustering performances of EPCM, PCM, PFCM and FCM are compared using three standard data sets. Section 2.4.8 illustrates a method to solve cluster parameters from parametrized **prototypes**. The cluster parameters characterises the shape and type of cluster structure developed. The chapter concludes with a brief review of the fuzzy partition space in Section 2.4.8 and the relationship between crisp and fuzzy cluster partitions.

2.4.1 Introduction

Fuzzy sets as a basis for clustering were first suggested by Bellman, Kalaba and Zadeh [1966] from which several classification schemes were developed [Gitman and Levine, 1970]. In 1969, Ruspini presented the first systematic account of fuzzy clustering [Ruspini, 1969]. Dunn developed the first fuzzy extension of the least squares clustering

criterion [Dunn, 1973]. Bezdek generalised this to an infinite family of algorithms [Bezdek, 1973].

Set membership can be realised mathematically by a membership function defined for each of the clusters as $u : X \rightarrow [0,1]$ where $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ is the data set and $u(\mathbf{x})$ gives the grade of membership of a feature vector $\mathbf{x} \in X$ in the fuzzy set u (see Section 2.4.9 for details). The memberships (eg. u_{ik}) are suffixed by two indices; i and k refer respectively to the row index (of the ith cluster) and column index (of the kth data element).

All fuzzy clustering objective functions use memberships as a weighting on the metric or norm. The fuzzy clustering algorithm is characterised by a minimisation of the objective function by a Picard iteration involving the memberships and prototypes, to generate **optimal** c partitions of the data in the sense of the least squared error criterion.

2.4.2 Ruspini's Objective Function

The objective function of Ruspini [1970] contains three clustering criteria. Denoting the objective function by J_R, the form is

$$J_R = \sum_{j=1}^{N} \sum_{k=1}^{N} \left\{ \left[\sum_{i=1}^{c} \sigma (u_{ij} - u_{ik})^2 \right] - d_{jk}^2 \right\}^2 \qquad (2.4.1)$$

where σ is a constant, u_{ij} and u_{ik} are memberships, and $2 \leq c < N$ is fixed a priori. The metric is denoted by d_{jk}. The optimal fuzzy c-partitions of a data set X is taken to be the local minima of J_R (global minima of prototypes are difficult to attain on account of the complex surface described by J_R). Ruspini considered J_R as a measure of cluster quality based on the local density because it will be small when the terms of (2.4.1) are each small. This occurs when close pairs of points have nearly equal membership. We will ignore the details of the algorithm (described in [Bezdek, 1981]), except to note a few features that have general significance for other fuzzy clustering algorithms. Specifically:

1. The stationary points of any objective function are not necessarily the local minima.
2. There is no assurance that a global optimum of an objective function gives a "good" clustering.
3. Different choices of the cluster parameters (eg. u and d) may lead to different "optimal" partitions.

4. A reasonable cluster structure may exist for more than one value of c.

These four points do not mean that useful cluster results are impossible to obtain. They do however, caution against uncritical interpretation of the cluster results, and emphasise the need for cluster validation. A practical solution is to use the algorithm which works best for the particular data set.

Figure 2.5, adapted from [Bezdek, 1981] shows the result of applying Ruspini's algorithm for two clusters ($c = 2$). Point membership value is indicated next to the circled point number. The significant point to note is that point number *8* forms a bridge between two symmetric clusters. Only the memberships of one cluster is shown on Figure 2.5. Memberships for the second cluster is symmetrically opposite to the first cluster. Under conventional crisp clustering, point number *8* must belong exclusively to one or the other cluster. However, this example shows that it is more natural to interpret point number *8* as belonging equally to both clusters, hence a membership of 0.5. The two **cluster prototypes** located at the wing tips of the clusters are a consequence of the clustering criterion of (2.4.1).

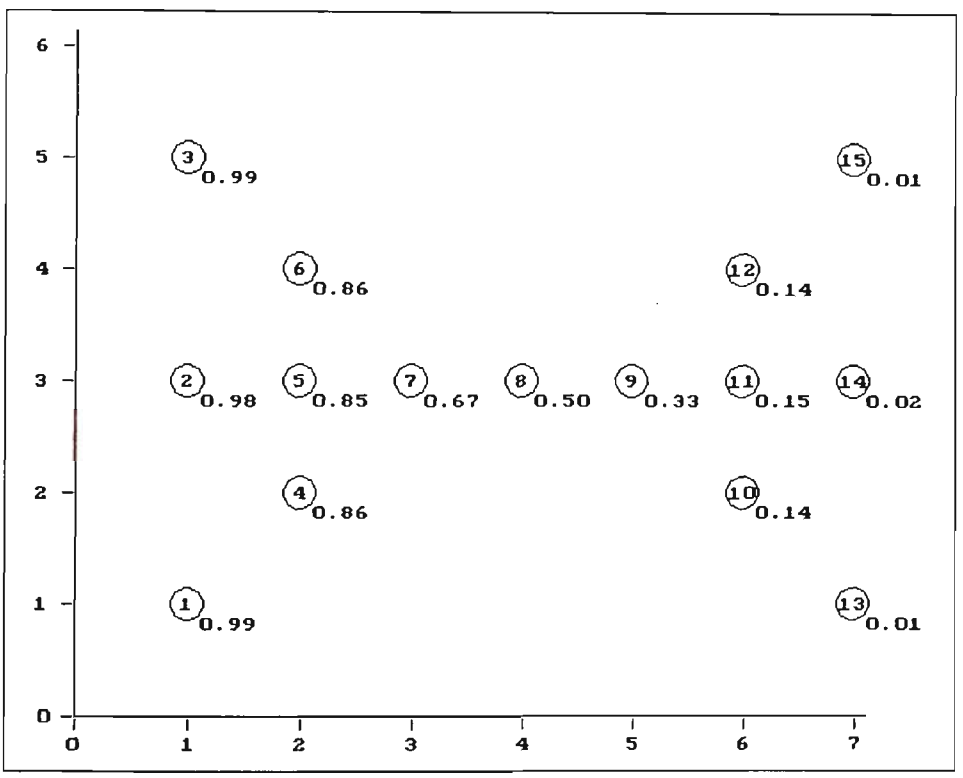


Figure 2.5 The butterfly membership assignment with Ruspini's algorithm. Only the memberships of one cluster is shown. Memberships of the second cluster is symmetrically opposite.

2.4.3 Hard c-Means (HCM)

The HCM has an objective function given by

$$J_h = \sum_{k=1}^{N} \sum_{i=1}^{c} u_{ik} d_{ik}^2 \tag{2.4.2}$$

where $d_{ik} = \|\mathbf{x}_k - \mathbf{v}_i\|$ is a Euclidean distance (Appendix A).

Note that J_h assesses the dissimilarity in d_{ik}, between $\mathbf{x}_k$ and $\mathbf{v}_i$ where $\mathbf{v}_i$ is not necessarily in X. Since $u_{ik} = u_i(\mathbf{x}_k) = 1$ if $\mathbf{x}_k$ belongs in ith cluster and is zero otherwise, (2.4.2) can be expressed as

$$J_h = \sum_{i=1}^{c} \sum_{\mathbf{x}_k \in u_i} \|\mathbf{x}_k - \mathbf{v}_i\|^2 \tag{2.4.3}$$

Minimising (2.4.2) with respect to $\mathbf{v}_i$ gives the cluster centroid

$$\mathbf{v}_i = \frac{\sum_{k=1}^{N} u_{ik} \mathbf{x}_k}{\sum_{k=1}^{N} u_{ik}} \tag{2.4.4}$$

If a scatter matrix for the ith cluster is defined as

$$S_i = \sum_{k=1}^{N} u_{ik} (\mathbf{x}_i - \mathbf{v}_k)(\mathbf{x}_i - \mathbf{v}_k)^T \tag{2.4.5}$$

and a within cluster scatter matrix as

$$S_h = \sum_{i=1}^{c} S_i \tag{2.4.6}$$

then if d_{ik} is defined by a Euclidean metric, it can be shown that the trace of S_h equals J_h of (2.4.2) given by

$$Tr(S_h) = J_h \tag{2.4.7}$$

where the trace of a $k \times k$ square matrix $A = \{a_{ij}\}$ is defined as the sum of its diagonal elements by

$$Tr(A) = \sum_{i=1}^{k} a_{ii}$$

From (2.4.3), J_h will be small when d_{ik} is small ie. when the points are close to their cluster centre. Consequently, J_h represents the overall within-group sum of squares errors over i. Since (2.4.3) is a measure of the squared Euclidean error in representing $\mathbf{x}_k$ by $\mathbf{v}_i$, J_h is also a measure of the local density. Hence the objective function is alternatively known as a density functional. From (2.4.7), the trace of S_h is proportional to the sum of variances. Therefore minimising J_h amounts to a minimisation of these variances. The preceding discussion illustrates that the objective function J_h has an appealing solution, from both geometric and statistical perspective. The HCM algorithm is obtained as the approximating minima of J_h by an iterative optimisation procedure given by the following algorithm:

The Hard *c*-Means (HCM) algorithm [Duda and Hart, 1973].

Step 1. Fix c, $2 \le c < N$, and initialise u_0.

Step 2. Calculate $\mathbf{v}_i$ from (2.4.4).

Step 3. Update memberships

$$u_{ik} = \begin{cases} 1, & d_{ik} = \min\limits_{1 \le j \le c} \{d_{jk}\} \\ 0, & \text{otherwise} \end{cases} \quad \text{for all } i \text{ and } k \tag{2.4.8}$$

Step 4. If $|U_t - U_{t-1}| < \varepsilon$, Stop, else go to Step 2.

Note: ε is a small number to control the stopping point and t is an iteration index.

Since the membership is hard (or crisp), $\mathbf{v}_i$ approximates to a cluster centroid. The condition of (2.4.8) is an assignment of points to the nearest cluster. If step four is replaced with the cluster centroids in the HCM algorithm, the KNN algorithm of Section 2.3.3 is obtained. Since the HCM algorithm is essentially a gradient descent technique, it is also sensitive to the four problem features noted for the Ruspini's algorithm.

2.4.4 ISODATA and Fuzzy *c*-Means

A more elaborate version of HCM is the ISODATA (acronym for Iterative Self Organising DAta Type A) algorithm by Ball and Hall [1967]. It is helpful to distinguish the ISODATA of Ball and Hall from the ISODATA of Bezdek. Bezdek's version is an extended form of Ball and Hall's ISODATA, by a generalisation of the membership exponent from $m = 2$ [Dunn, 1973] to $m = \infty$ [Bezdek, 1981].

The result of HCM is shown for one cluster memberships in Fig. 2.6. Memberships for the second cluster is symmetrically opposite. The cluster prototypes for v_1 and v_2 are (1.71, 3) and (6, 3) respectively. Comparing Fig. 2.5 with Fig. 2.6, it is evident that Ruspini's fuzzy partition conveys more information than the hard partition. The low membership of point number *8* signals a closer look at the data. The hard assignment of cluster points not only distorts the symmetry of cluster structures but also removes points that signify a problem condition, like point number *8*.

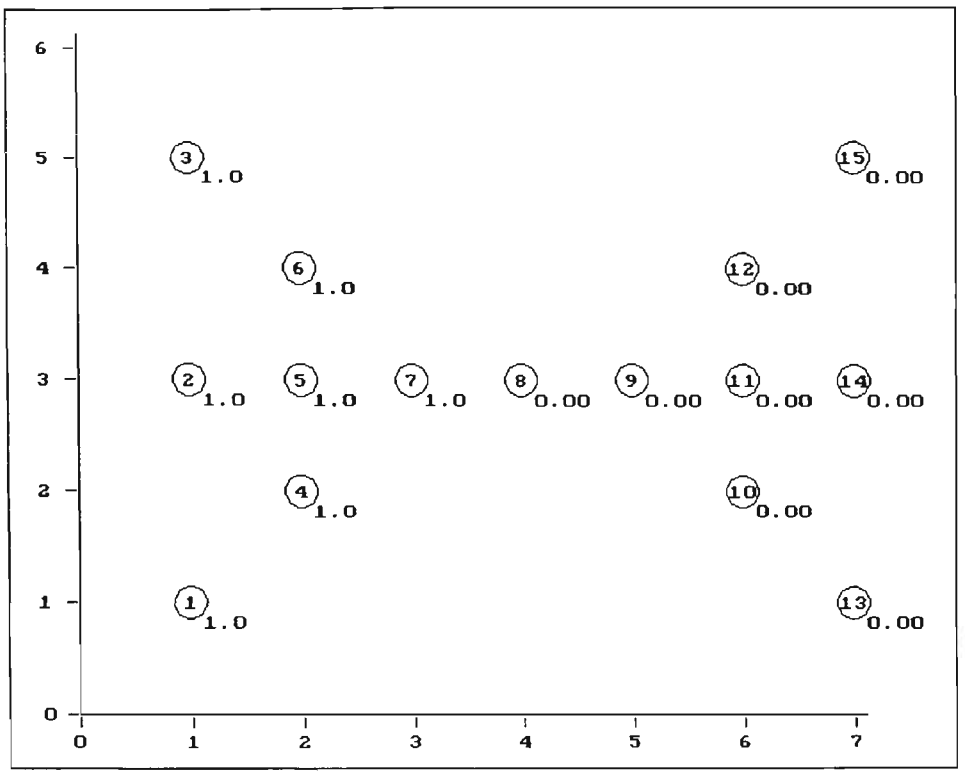


Figure 2.6 The butterfly memberships of HCM algorithm.

The generalisation of HCM objective function J_h was reported by Dunn [1973] which Bezdek subsequently extended to include a family of fuzzy clustering algorithms based on the least squared errors criterion [1973]. The generalisation of FCM is taken a step further with Bezdek and Im's PFCM to improve the definition cluster properties. Prior to doing this, we examine the analytic structure of FCM to elucidate similarity of form. The FCM algorithm is an iterative procedure for approximately minimising the objective function by a Picard iteration via the memberships and prototypes.

Fuzzy c-Means [Bezdek, 1981]. **Theorem** 2.4.1:

Let the data set X be defined as $X = \{\mathbf{x}_1,\mathbf{x}_2,...,\mathbf{x}_k,...\mathbf{x}_N\}$ for N items in the finite subset of $\Re^d$. Let the fuzzy objective function be defined as

$$J_m(U,V) = \sum_{k=1}^{N}\sum_{i=1}^{c} u_{ik}^m d_{ik}^p \qquad (2.4.9)$$

where $p = 2$ (for the case of FCM) and $d_{ik} = \|\mathbf{x}_k - \mathbf{v}_i\|_{A_i} = \sqrt{(\mathbf{x}_k - \mathbf{v}_i)^T A_i (\mathbf{x}_k - \mathbf{v}_i)}$ is an inner product norm weighted with a $d \times d$ positive definite matrix A_i. Let U defines the real memberships u_{ik} on X of $\mathbf{x}_k$ in the ith fuzzy subset, and V denotes the c-tuples of cluster prototypes, $\mathbf{v}_i$. for $i = 1,..,c$. Let the $c \times N$ matrix $U = [u_{ik}]$ be a constrained fuzzy c-partitions of X which satisfies three conditions:

$$u_{ik} \in [0,1] \quad \forall i,k, \quad 0 < \sum_{k=1}^{N} u_{ik} < N \quad \forall i \quad \text{and} \quad \sum_{i=1}^{c} u_{ik} = 1 \quad \forall k \tag{2.4.10}$$

Fix the fuzzifier $m \in (1,\infty)$ and let X have at least $c < N$ distinct points. Define for all k, the sets

$$I_k = \{i | 1 \le i \le c;\ d_{ik} = \|\mathbf{x}_k - \mathbf{v}_i\|_{A_i} = 0\}$$

$$\tilde{I}_k = \{1,2,...,c\} - I_k$$

then $J_m(U,V)$ may be globally minimised only if

$$I_k = \varnothing \Rightarrow u_{ik} = \frac{1}{\sum_{j=1}^{c} \left(\frac{d_{ik}^2}{d_{jk}^2} \right)^{\frac{1}{(m-1)}}} \tag{2.4.11}$$

$$I_k \neq \varnothing \Rightarrow u_{ik} = 0, \forall i \in \tilde{I}_k \quad \text{and} \quad \sum_{i \in I_k} u_{ik} = 1 \tag{2.4.12}$$

$$\mathbf{v}_i = \frac{\sum_{k=1}^{N} u_{ik}^m \mathbf{x}_k}{\sum_{k=1}^{N} u_{ik}^m} \tag{2.4.13}$$

The proof of Theorem 2.4.1 is given in Appendix B. Convergence theorems for FCM exist [Bezdek, 1981]. The FCM algorithm is presented below:

FCM algorithm [Bezdek, 1981].

Step 1. Fix c, $2 \le c < N$. Choose any inner product norm defined by d_{ik} in (2.4.9).

Step 2. Fix m. Normally $m = 2$ is satisfactory. Initialise U_0 (or V_0).

Step 3. Calculate prototypes $\mathbf{v}_i$ from (2.4.13) with u_{ik} from (2.4.11).

Step 4. Update u_{ik} from (2.4.11) with $\mathbf{v}_i$ from (2.4.13).

Step 5. If $|U_t - U_{t-1}| < \varepsilon$, Stop, else go to Step 3.

Note: ε is a small number to control the stopping point and t is the iteration index.

There are several features of FCM to be observed. Initialisation of the memberships can be satisfactorily accomplished with random numbers less than unity. Hall et al. [1992] gives another method consisting of pairs of 1s, offset sequentially by two positions along each subsequent row. It is also possible to use random initial prototypes, V_0. The occurrence of singularity in (2.4.11) is avoided by specifying a cluster set I_k. If singularity occurs ($d_{ik} = 0$), then the membership is defined by (2.4.12). The metric d_{ik} can include any

inner product norm with a symmetric positive definite weighting matrix A. This type of norm is useful for detecting non-circular clusters such as the elliptic or linear cluster varieties. An example of its use is given in Section 3.4.2. FCM has the interesting property where $m \rightarrow 1^+$ results in crisp memberships in which $u_{ik} \in \{0,1\}$. The crisp memberships may also be obtained from

$$u_{ik} = \begin{cases} 1 & d_{ik} = \min\limits_{1<j\le c}\{d_{jk}\} \\ 0 & \text{otherwise} \end{cases} \tag{2.4.14a}$$

for $1 \le i \le c$ and $1 \le k \le N$. Alternatively, crisp partitions may be obtained from fuzzy partitions by applying the maximum membership rule, ie. selecting a point with the maximum membership from all c clusters from the relation

$$u_{ik} = \begin{cases} 1 & u_{ik} = \max\limits_{1<j\le c}\{u_{jk}\} \\ 0 & \text{otherwise} \end{cases} \tag{2.4.14b}$$

The FCM algorithm contains a number of parameters, the more important ones being c and m, denoting the number of clusters (for $c > 1$) and the "smoothing" or "fuzziness" factor m. There are no analytic relations to calculate m. The best guide is to select the value of m that fits the data best. Normally, a value of two or three is satisfactory. In FCM, $\mathbf{v}_i$ is dependent on m (see Eqs. 2.4.11 and 2.4.13). FCM solutions are restricted to $m > 1$. As m approaches infinity, the memberships in all clusters tend towards a value of $1/c$. This result is a consequence of the membership constraint of (2.4.10).

Figure 2.7 shows the FCM memberships for one fuzzy cluster of Ruspini's butterfly data with $m = 1.25$. Memberships for the second cluster are symmetrically opposite. For this case, the partitions are considered hard, but the bridge point number *8* preserves the fuzzy membership. However, decreasing m close to unity will produce the totally hard clusters of Fig. 2.6. Figure 2.8 shows the memberships with $m = 2$. Note the effect of m on the stationary points of the cluster prototypes (compare Figs. 2.7 and 2.8).

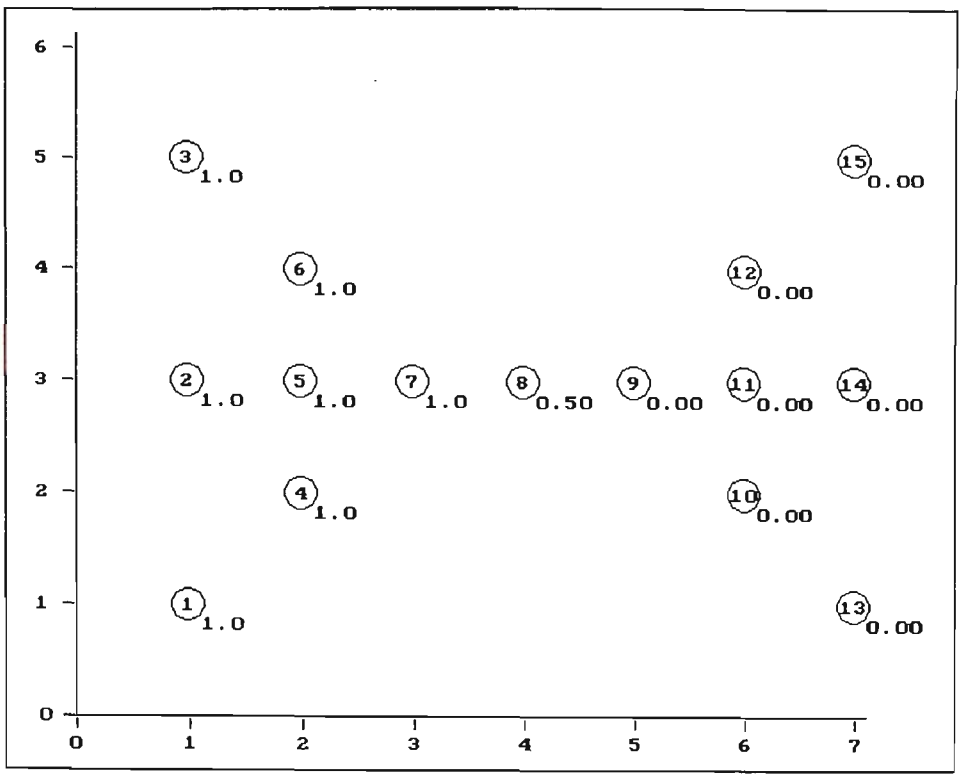


Figure 2.7 The butterfly memberships of FCM algorithm, using $m = 1.25$. Cluster prototypes are located at (1.843, 3) and (6.155, 3). Only the memberships of one cluster at (1.843,3) is shown.

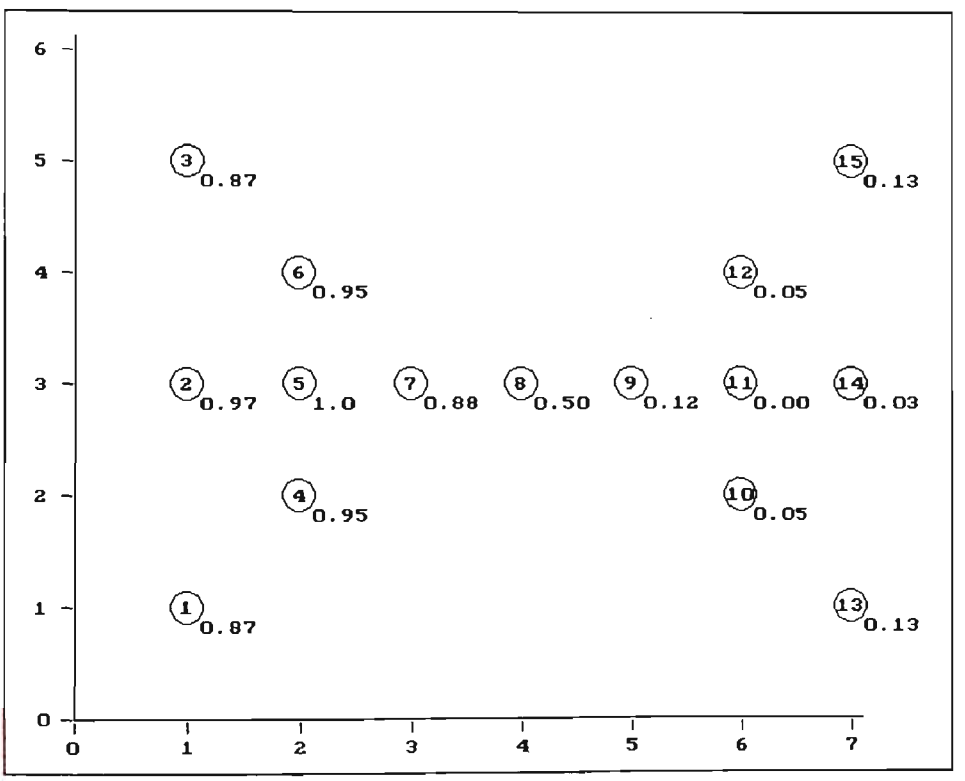


Figure 2.8 The butterfly memberships of FCM algorithm, using $m = 2$. Cluster prototypes are located at (1.855, 3) and (6.145, 3). Only the memberships of one cluster at (1.855,3) is shown.

2.4.5 Possibilistic Fuzzy *c*-Means (PFCM)

The PFCM extends FCM in two ways. Firstly, it generalises the metric d_{ik} with an exponent p. Secondly, it generalises the membership constraint with an exponent α (see Eq. (2.4.16)).

Possibilistic Fuzzy *c*-Means (Bezdek and Im). **Theorem** 2.4.2:

For similar notations and parameters to FCM, define the objective function as

$$J_{m,p,\alpha}(U,V)=\sum_{k=1}^{N}\sum_{i=1}^{c}u_{ik}^{m}d_{ik}^{p} \tag{2.4.15}$$

where

$$d_{ik}=\|\mathbf{x}_k-\mathbf{v}_i\|_A=\sqrt{(\mathbf{x}_k-\mathbf{v}_i)^T A(\mathbf{x}_k-\mathbf{v}_i)}$$

represents the weighted distance of a feature vector (or data point) $\mathbf{x}_k$ from prototype $\mathbf{v}_i$, with a $d \times d$ symmetric positive definite matrix A. Let the fuzzy c-partitions of N items of X in $\Re^d$ satisfy the following three conditions:

$$u_{ik}\in[0,1] \quad \forall i,k, \quad 0<\sum_{k=1}^{N}u_{ik}<N \quad \forall i \quad \text{and} \quad \sum_{i=1}^{c}u_{ik}^{\alpha}=1 \quad \forall k,\alpha \tag{2.4.16}$$

Define for all k, the sets

$$I_k=\{i \mid 1\le i\le c;\ d_{ik}=\|\mathbf{x}_k-\mathbf{v}_i\|=0\}$$

$$\tilde{I}_k=\{1,2,\ldots,c\}-I_k$$

then $J_{m,p,\alpha}(U,V)$ may be globally minimised only if

$$I_k=\varnothing \Rightarrow u_{ik}=\frac{1}{\left[\sum_{j=1}^{c}\left(\frac{d_{ik}^{p}}{d_{jk}^{p}}\right)^{\frac{\alpha}{m-\alpha}}\right]^{\frac{1}{\alpha}}} \tag{2.4.17}$$

$$I_k\neq\varnothing \Rightarrow u_{ik}=0,\forall i\in\tilde{I}_k \quad \text{and} \quad \sum_{i\in I_k}u_{ik}=1 \tag{2.4.18}$$

$$\mathbf{v}_i=\frac{\sum_{k=1}^{N}u_{ik}^{m}d_{ik}^{p-2}\mathbf{x}_k}{\sum_{k=1}^{N}u_{ik}^{m}d_{ik}^{p-2}}, \quad 1\le i\le c \tag{2.4.19a}$$

Relaxing the minimisation condition for $J_{m,p,\alpha}$ with respect to $\mathbf{v}_i$, improved computation efficiency and convergence around centroids may be obtained from the alternative solution

$$\mathbf{v}_i = \frac{\sum_{k=1}^{N} u_{ik}^{\frac{m}{p-1}} \mathbf{x}_k}{\sum_{k=1}^{N} u_{ik}^{\frac{m}{p-1}}} \qquad (2.4.19b)$$

Equation (2.4.19b) does not involve the metric d_{ik}, hence it is algorithmically simpler and more efficient to compute. Its structure suggests that prototype development is governed by the fuzzy and metric exponents m and p respectively, to provide greater control over centroid clustering. The metric exponent in (2.4.19b) is resticted to $p > 1$ so that its unit norm lies on the upper triangle of Fig. 2.2. However in (2.4.19a), the exponent $p > 0$ so that its unit norm lies within the square boundary.

Both prototype equations show no convergence problems in the normal range of the parameters. The proof of PFCM is given in Appendix C. Note that the forms of (2.4.17) and (2.4.19) are similar to (2.4.11) and (2.4.13), except for the extra exponent terms, α and p. The algorithm steps involving (2.4.19a) and (2.4.19b) are slightly different. For identification purpose, these are labelled as PFCM1 and PFCM2:

PFCM1 algorithm with equation (2.4.19a), (Bezdek and Im).

Step 1. Fix c, $2 \le c < N$. Choose any inner product norm defined by d_{ik} in (2.4.15).

Step 2. Fix α, m and p. Normally $m = 2$ and $p = 2$ are satisfactory. Initialise U_0 (or V_0).

Step 3. Calculate initial prototypes from (2.4.13).

Step 4. Update memberships u_{ik} from (2.4.17).

Step 5. Update prototypes $\mathbf{v}_i$ from (2.4.19a).

Step 6. If $|U_t - U_{t-1}| < \varepsilon$, Stop, else go to Step 4.

PFCM2 algorithm with equation (2.4.19b), (Bezdek and Im).

Step 1. Fix c, $2 \le c < N$. Choose any inner product norm defined by d_{ik} in (2.4.15).

Step 2. Fix α, m and p. Normally $m = 2$ and $p = 2$ are satisfactory. Initialise U_0 (or V_0).

Step 3. Update prototypes $\mathbf{v}_i$ from (2.4.19b) with u_{ik} from (2.4.17).

Step 4. Update memberships u_{ik} from (2.4.17) with $\mathbf{v}_i$ from (2.4.19b).

Step 5. If $|U_t - U_{t-1}| < \varepsilon$, Stop, else go to Step 3.

Note: ε is a small number to control the stopping point and t is an iteration index.

The parameter α is an exponent of the membership, generalised from the third condition of (2.4.10) to the third condition of (2.4.16). It may be verified that PFCM reverts to FCM for parameter values of $p = 2$ and $\alpha = 1$.

The substitution of $\alpha = m/2$ in (2.4.17), yields an interesting result for the membership function given by

$$u_{ik} = \frac{1}{\left[\sum_{j=1}^{c}\left(\frac{d_{ik}^{p}}{d_{jk}^{p}}\right)^{\frac{2}{m}}\right]} \tag{2.4.20}$$

Recall that FCM's membership equation (2.4.11) is restricted to m > 1. In (2.4.20), the range of m is extended to $m > 0$. Another useful result is obtained by substituting (2.4.20) into (2.4.19). Thus unlike FCM, the special case of PFCM with $\alpha = m/2$ produces prototypes that are independent of m.

It may also be noted that the extreme values of the α parameter represent a more general expression of memberships compared to the two cases of FCM. For example, the crisp case of FCM as $m \rightarrow 1^{+}$ ($m = 1$ is undefined), corresponds to $\alpha \rightarrow m$ for $0 < m < \infty$ of PFCM. Likewise, the extremely fuzzy case of FCM for $m \rightarrow \infty$, corresponds to $\alpha \approx 0$ for $0 < m < \infty$ of PFCM. In other words, each extreme condition of FCM can be represented by a large family of memberships involving m and p. The implications of these possibilities are yet to be explored.

The intermediate range $0 < \alpha < m$ of PFCM for $0 < m < \infty$, give a more extensive description of the memberships than is available from FCM. The possible PFCM membership functions are depicted in Fig. 2.9(a) to 2.9(h). These distributions are obtained from the data set of Kaufman and Rousseuw shown on Fig. 2.11. The memberships represent cluster number *1* at (7.001, 2) calculated for $x = 7$ and $y = 1$ to 14. The cluster centres for the other two clusters were assumed to be (2.001, 9) for cluster *2* and (13.501, 9) for cluster *3*. In calculating the membership distribution, each x-coordinate position of the prototype was made to deviate slightly (by 0.001) from its true position to avoid singularity in the d_{ik} term. Figure 2.10(a) to 2.10(d) show the FCM's membership functions for the same cluster prototypes.

A comparison of the FCM and PFCM membership distributions reveals some similarities and differences. The similarities are: (i) same y-coordinate crossover point of the curves at $y = 7.5$ units, (ii) typical fuzzy profile (bell shape) and crisp profile (hat shape), and (iii) membership equality at parameters $m = 2$ and $\alpha = 0.5m$. The differences are: (i) the crossover points in FCM have a fixed membership that is inversely proportional to the number of clusters, whereas PFCM has a range of crossover points, depending on the alpha value, and (ii) FCM has no equivalent functions similar to Figs. 2.9(a) and (b).

There are some interesting features in the curves of Figs. 2.9 and 2.10. The crossover point represents the decision boundary at which a feature vector has equal membership in each of the three clusters of Fig. 2.11. This condition occurs when $d_{ik} = d_{jk}$. Membership at this boundary for the case of FCM is

$$u_{bp} = \frac{1}{c} \tag{2.4.21}$$

and for PFCM, it is

$$u_{bp} = \frac{1}{c^{1/\alpha}} \tag{2.4.22}$$

where c is the number of clusters and the subscript bp denotes a boundary point.

For the three clusters of Fig. 2.11, FCM has $u_{bp} = 0.33$ (to two decimal places). In PFCM, α (in Eq. (2.4.22)) may be adjusted so that $u_{bp} = 0.5$ gives a convenient value that conforms to the convention used to represent a decision boundary in conventional or neural classifiers.

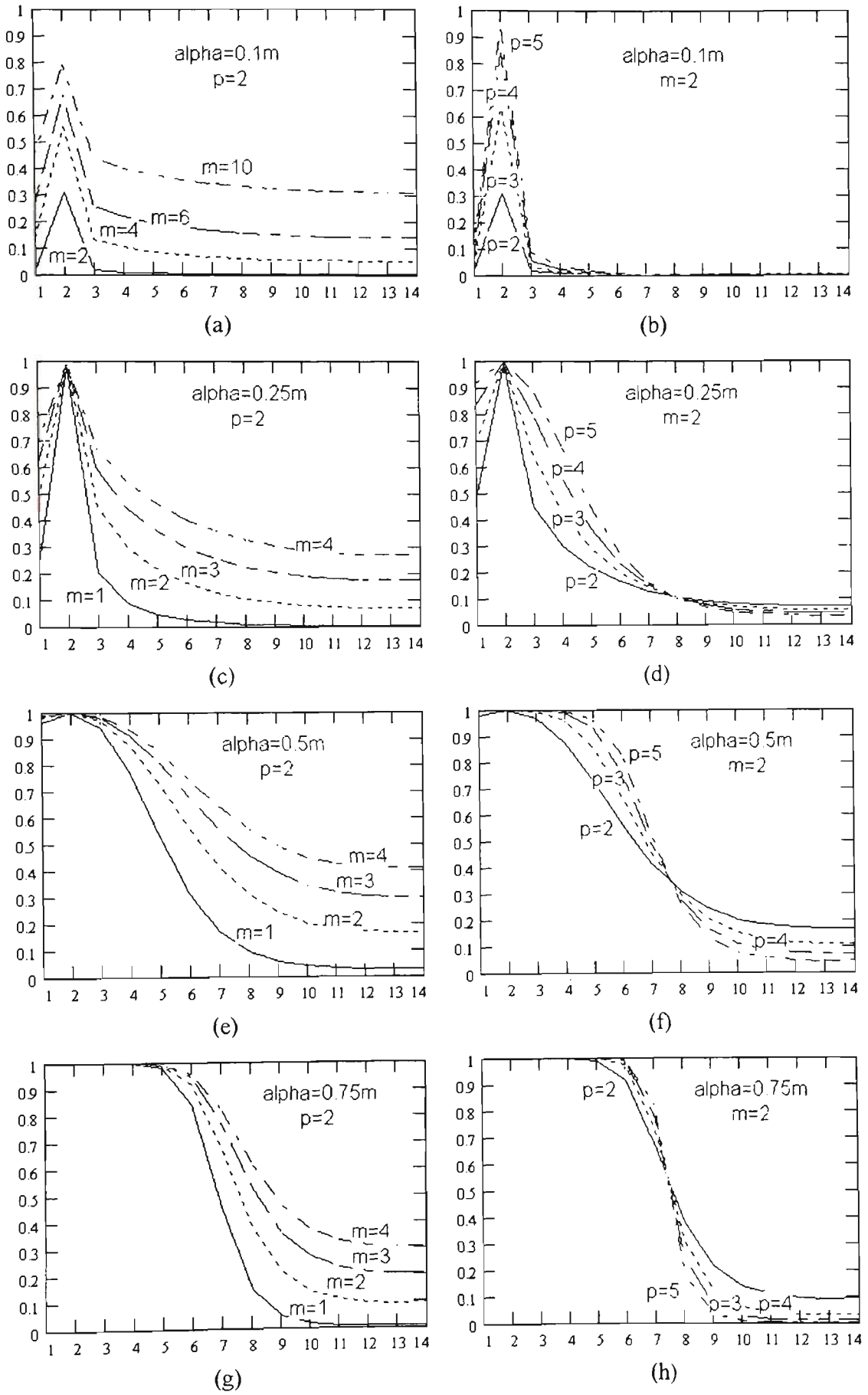


Figure 2.9 PFCM membership functions for data set of Fig. 2.11. Cluster centre is at (7, 2) and memberships are calculated along $x = 7$ for $y = 1$ to $y = 14$.

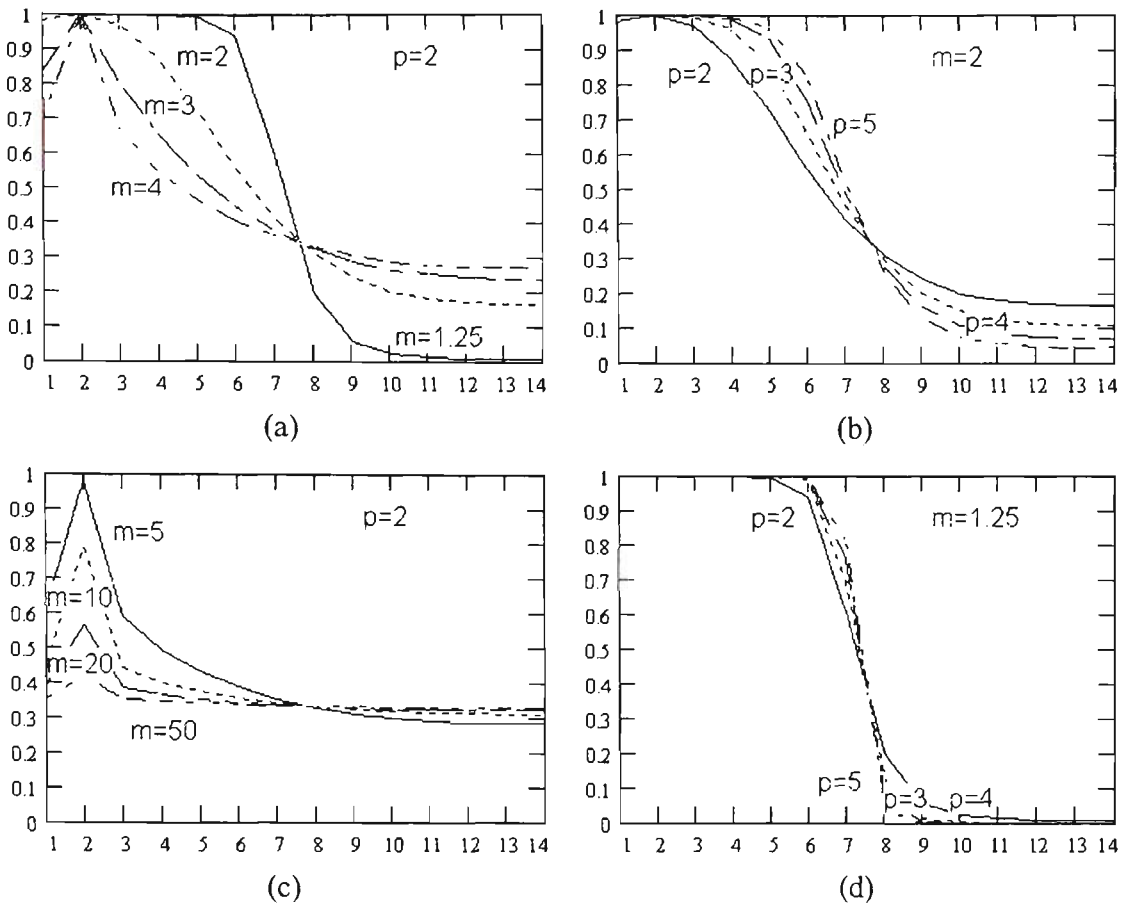


Figure 2.10 FCM membership functions for data set of Fig. 2.11. Cluster centre is at (7, 2) and memberships are calculated along $x = 7$ for $y = 1$ to $y = 14$.

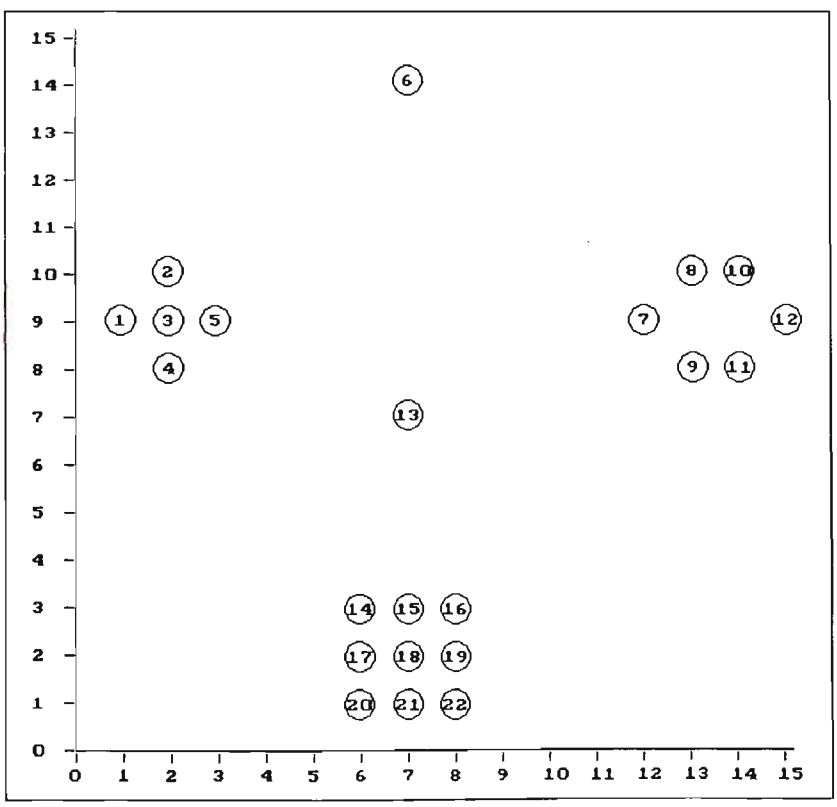


Figure 2.11 Data set from [Kaufman and Rousseeuw, 1990]. Each point is labelled by a number. The coordinates of each point is located at the centre of the labelled circle.

2.4.5.1 Comparing Fuzzy Clusters at Centroids

Table 2.3 compares the best centroid clustering results of PFCM1, PFCM2 and FCM, using data from Fig. 2.11. The cluster parameters of $m = 3.17$ and $\alpha = 0.5m$ for PFCM2 produce a decision boundary at $u_{bp} = 0.5$. Points *6* and *13* have interesting fuzzy values. For the specified boundary membership, point *6* is seen to belong to clusters *1* and *3*, and point *13* to clusters *1* and *2*. Both PFCM and FCM show point *13* with a higher membership in cluster *2*. These results agree with expectation. The good cluster points (shaded) of FCM have a few memberships less than 0.9, whereas all memberships of PFCM exceed 0.9. This result indicates that PFCM identifies clusters more positively than FCM. Overall, PFCM gives a slightly better assessment of the cluster result. Memberships of PFCM1 and PFCM2 are almost similar. In this example, the cluster prototypes are centred at the local cluster centroid (such prototypes are alternatively referred to as centred prototypes). Clustering at the local cluster centroids is a characteristic of the clustering criterion defined by the objective function and determined by the parameters α, m and p. Table 2.3 shows both FCM and PFCM exhibit centroid clustering for the selected parameter values. However, note that for $p = 2$ at normal values of m, FCM clusters poorly at centroids.

k	u_1			u_2			u_3		
	PFCM1	PFCM2	FCM	PFCM1	PFCM2	FCM	PFCM1	PFCM2	FCM
1	.985	.961	.943	.692	.125	.035	.660	.094	.022
2	.985	.967	.944	.688	.110	.032	.667	.091	.024
3	1	.999	1	.345	.013	.0	.329	.010	0
4	.983	.954	.932	.708	.144	.043	.666	.099	.024
5	.983	.961	.934	.702	.125	.039	.675	.097	.027
6	.891	.603	.442	.818	.361	.199	872	.519	.360
7	.711	.156	.047	.729	.178	.058	.975	.924	.895
8	.673	.108	.027	.684	.118	.031	.985	.960	.942
9	.681	.113	.030	.710	.144	.045	.981	.950	.925
10	.674	.099	.027	.688	.111	.032	.984	.964	.941
11	.680	.104	.030	.710	.133	.044	.981	.956	.926
12	.700	.122	.040	.723	.147	.054	.978	.946	.907
13	.869	.523	.354	.880	.556	.393	.839	.415	.253
14	.749	.198	.076	.971	.918	.874	.716	.150	.050
15	.706	.137	.042	.981	.950	.923	.692	.122	.035
16	.730	.170	.060	.972	.920	.877	.735	.176	.064
17	.703	.133	.041	.982	.956	.931	.677	.105	.029
18	.238	.002	.0	1	1	1	.234	.002	0
19	.689	.117	.033	.982	.957	.931	.693	.120	.035
20	.727	.162	.057	.976	.937	.901	.704	.134	.042
21	.687	.114	.033	.984	.962	.939	.678	.104	.029
22	.714	.146	.048	.977	.938	.902	.718	.150	.050

Table 2.3 Comparison of fuzzy clusters at centroids. The symbols u_1, u_2 and u_3 represent memberships of cluster *1*, *2* and *3* respectively. Parameters and prototypes associated with each algorithm are listed as Type = (α, m, p, v_1, v_2, v_3): PFCM1 = (7.5, 15, 1.2, (2.01, 9), (7, 2), (13.39, 9.05)), PFCM2 = (1.585, 3.17, 1.5, (2.07, 9.06), (7, 1.99) (13.51, 9.04)) and FCM = (1, 1.8, 1.2, (2 03,9.01), (7, 2), (13.39, 9.05)). True

cluster centroids are at: $v_1 = (2, 9)$, $v_2 = (7, 2)$ and $v_3 = (13.5, 9)$. Memberships of the three clusters are shown shaded. The data set is from Fig. 2.11.

2.4.5.2 Comparing Cluster Width Selectivity at Centroids

Table 2.4 compares the cluster width selectivity of PFCM with FCM, at best centroids. Cluster width selectivity refers to the ability of the algorithm to select a small core of the cluster that is centred at the cluster prototype. In the case of FCM, the core of cluster width is fixed (see Eq. (2.4.21)). Consequently, the dynamic range of the fuzzy memberships is limited by $u_{bp} = 0.33$, for three clusters. To apply an alphacut (or membership threshold) on the FCM's memberships, one needs to know the function's characteristics such as Fig. 2.10(a). PFCM offers a simpler procedure for selecting the alphacut. If we select the parameter α to correspond to a desired cluster width, then the cluster core is extracted for memberships above a threshold u_{co}. Table 2.4 shows that $u_{co} \geq 0.05$ selects a cluster core radius of about 1.5 units of the axes scale of Fig. 2.11. This occurs for low alpha values ($\alpha < 0.3$). The boundary at $u_{bp} = 0.012$ is less than u_{co} by a factor of 4. A low α permits setting a low u_{co} threshold for the extraction of the cluster core (and a lower u_{bp}). In the case of FCM, it is more difficult to establish a lower limit for u_{co}.

k	u_1			u_2			u_3		
	PFCM1	PFCM2	FCM	PFCM1	PFCM2	FCM	PFCM1	PFCM2	FCM
1	.215	.182	.630	0	.001	.200	0	0	.171
2	.247	.186	.634	0	.001	.192	0	0	.174
3	.570	.721	.876	0	0	.066	0	0	.057
4	.215	.165	.622	.001	.002	.208	0	0	.170
5	.262	.169	.627	0	.001	.199	0	0	.174
6	.022	.020	.373	.005	.006	.280	.015	.014	.347
7	0	.002	.199	.001	.002	.215	.170	.114	.586
8	0	0	.179	0	0	.188	.147	.183	.633
9	0	.001	.178	0	.002	.205	.208	.141	.617
10	0	0	.182	0	0	.195	.234	.197	.623
11	0	0	.182	.001	.002	.209	.203	.154	.609
12	0	.001	.197	.001	.002	.219	.174	.145	.583
13	.014	.014	.342	.017	.016	.354	.007	.008	.304
14	.002	.003	.235	.150	.110	.562	0	.001	.203
15	0	.001	.202	.216	.151	.608	0	.001	.190
16	.001	.002	.216	.152	.111	.562	.001	.002	.222
17	0	.001	.203	.216	.163	.617	0	0	.180
18	0	0	.033	.729	.847	.935	0	0	.032
19	0	.001	.189	.220	.164	.618	0	.001	.193
20	.001	.002	.220	.166	.131	.583	0	.001	.198
21	0	.001	.191	.221	.177	.626	0	0	.183
22	.001	.002	.207	.168	.131	.583	.001	.002	.210

Table 2.4 Comparison of cluster width selectivity at centroids. The symbols u_1, u_2 and u_3 represent memberships of cluster *1*, *2* and *3* respectively. Parameters and prototypes associated with each algorithm are listed as Type = (α, m, p, v_1, v_2, v_3.): PFCM1 = (0.25, 1, 2, (2.21, 9.04), (7.01, 2.07), (13.34, 9.07)),

PFCM2 = (0.25, 1, 1.7, (2.03, 9.02), (7, 1.99) (13.52, 9.15)) and FCM = (1, 4, 1.6, (2.07,9), (7, 2.02), (13.31, 9.02)). True cluster centroids are at: $v_1 = (2, 9)$, $v_2 = (7, 2)$ and $v_3 = (13.5, 9)$. Memberships of the three clusters are shown shaded. The data set is from Fig. 2:11.

Generally, PFCM1 and PFCM2 possess higher membership resolution, hence superior cluster width selectivity than FCM. Like the case of fuzzy clusters, PFCM1 and PFCM2 have nearly similar memberships. For the selected parameters, all algorithms show a possibilistic membership distribution, including FCM.

2.4.5.3 Comparing Crisp Clusters

The crossover point has other interesting implications for clustering. Figure 2.9 suggests that it is possible to select the cluster "bandwidth" by adjusting m and using α to tune the "quality factor" or selectivity. For example, if one is interested in capturing a representative portion of the fuzzy cluster data, one possible set of cluster parameters to use is $\alpha \approx 0.08$, $m \approx 0.1$ and $p \approx 1$. These parameters have the effect of isolating very fuzzy points so that only representative data are contained in the cluster. This feature is only approximated in FCM by applying an alphacut to the fuzzy membership, since all points are fully assigned in the crisp partitioning case.

Crisp partitions of the Kaufman and Rousseeuw's data set and Krishnapuram and Keller's data set are shown in Table 2.7 and 2.8 respectively. The position of each point of Krishnapuram and Keller's data set is given in Table 2.6. The intermediate noise points numbered *6* and *13* (see Fig. 2.11), shown shaded in Table 2.7, are incorrectly assigned to the class of cluster number *1* (with v_1) for the case of FCM. PFCM gives the best cluster interpretation by not assigning these points to any clusters. In Table 2.8, FCM forces the noise points numbered *1* and *2* (see Fig. 2.12) into the number *2* cluster (with v_2). PFCM gives these points zero membership, thus correctly identifying noise points. Note that a normal $p = 2$ is used to calculate cluster prototypes for FCM (also recall that $\alpha = 1$ and $p = 2$ for PFCM is equivalent to Bezdek's FCM). Consequently, the prototypes are not accurately located at cluster centroids which is a reason for the poor cluster result.

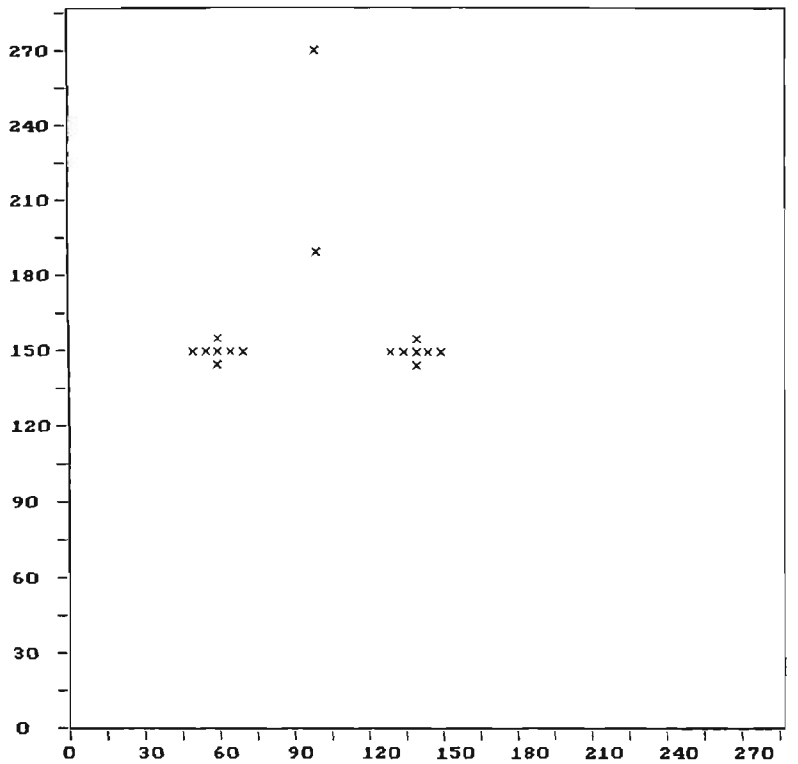


Figure 2.12 Krishnapuram and Keller's data set (see Table 2.6) with noise point number *1* at (100, 270) and noise point number *2* at (100, 190) [Krishnapuram and Keller, 1993]. Noise points *1* and *2* are denoted by the two topmost points of the figure.

Point	Coordinate	Point	Coordinate
1	60,155	8	130,150
2	140,155	9	135,150
3	50,150	10	140,150
4	55,150	11	145,150
5	60,150	12	150,150
6	65,150	13	60,145
7	70,150	14	140,145

Table 2.5 Point positions of Krishnapuram and Keller's data set without noise [1993].

Point	Coordinate	Point	Coordinate
1	100,270	9	70,150
2	100,190	10	130,150
3	60,155	11	135,150
4	140,155	12	140,150
5	50,150	13	145,150
6	55,150	14	150,150
7	60,150	15	60,145
8	65,150	16	140,145

Table 2.6 Point positions of Krishnapuram and Keller's data set with noise [1993]. Noise points number *1* and *2* (see Fig. 2.12) are shown shaded.

Point	Membership			Point	Membership		
	PFCM1	PFCM2	FCM		PFCM1	PFCM2	FCM
1	1,0,0	1,0,0	1,0,0	12	0,0,1	0,0,1	0,0,1
2	1,0,0	1,0,0	1,0,0	13	0,0,0	0,0,0	.93, 07,0
3	1,0,0	1,0,0	1,0,0	14	0,1,0	0,1,0	0,1,0
4	1,0,0	1,0,0	1,0,0	15	0,1,0	0,1,0	0,1,0
5	1,0,0	1,0,0	1,0,0	16	0,1,0	0,1,0	0,1,0
6	03,0,0	02,0,0	1,0,0	17	0,1,0	0,1,0	0,1,0
7	0,0,1	0,0,1	0,0,1	18	0,1,0	0,1,0	0,1,0
8	0,0,1	0,0,1	0,0,1	19	0,1,0	0,1,0	0,1,0
9	0,0,1	0,0,1	0,0,1	20	0,1,0	0,1,0	0,1,0
10	0,0,1	0,0,1	0,0,1	21	0,1,0	0,1,0	0,1,0
11	0,0,1	0,0,1	0,0,1	22	0,1,0	0,1,0	0,1,0

Table 2.7 Crisp partitioning of Kaufman and Rousseeuw's data set (Fig. 2.11) into three clusters. Parameters and prototypes associated with each algorithm are listed as Type = (α, m, p, v_1, v_2, v_3): PFCM1 = (0.09, 0.1, 0.8, (2, 9), (7, 2), (13.44, 9.06)), PFCM2 = (0.085, 0.1, 1.1, (2.02, 9.02), (7, 1.99), (13.5, 9)) and FCM = (1, 1.1, 2, (3.39, 9.46), (7, 2.03), (13.5, 9)). True cluster centroids are at: v_1 = (2, 9), v_2 = (7, 2) and v_3 = (13.5, 9). The data set is from Fig. 2.11. Note the zero memberships of points *6* and *13* for PFCM1 and PFCM2 (shown shaded).

Point	Membership			Point	Membership		
	PFCM1	PFCM2	FCM		PFCM1	PFCM2	FCM
1	0,0	0,0	0,1	9	1,0	1,0	1,0
2	0,0	0,0	0,1	10	0,1	0,1	0,1
3	1,0	1,0	1,0	11	0,1	0,1	0,1
4	0,1	0,1	0,1	12	0,1	0,1	0,1
5	1,0	1,0	1,0	13	0,1	0,1	0,1
6	1,0	1,0	1,0	14	0,1	0,1	0,1
7	1,0	1,0	1,0	15	1,0	1,0	1,0
8	1,0	1,0	1,0	16	0,1	0,1	0,1

Table 2.8 Crisp partitioning of Krishnapuram and Keller's data set (Fig. 2.12 and Table 2.6) into two clusters. Parameters and prototypes associated with each algorithm are listed as Type = (α, m, p, v_1, v_2): PFCM1 = (0.09, 0.1, 0.8, (60, 150), (140, 150)), PFCM2 = (0.085, 0.1, 1.1, (60, 150), (140, 150)) and FCM = (1, 1.05, 2, (60, 150.01), (131.12, 167.76)). True cluster centroids are at: v_1 = (60, 150), v_2 = (140, 150). Note the zero memberships of points *1* and *2* for PFCM1 and PFCM2 (shown shaded).

PFCM offers some convenience features like improved range of m selection, prototype equations that are independent of m, improved cluster width selectivity, automatic isolation of very fuzzy points, and an easier, more natural way to define boundary points. More importantly, both PFCM and FCM link an objective function to cluster characterisation. The parameters of the objective function yield analytic solutions that can be optimised for different conditions of the cluster structure. Thus, by imposing specific constraints on

the metric d_{ik}, the algorithm can be made to detect ellipse shape clusters, such as the Gustafson-Kessel algorithm [1979] with fuzzy covariance matrix.

2.4.6 Objective Function with Variable Norms

Gustafson and Kessel's algorithm is significant in that it represents a generalisation of the norm to detect a diversity of hyperellipsoidal cluster substructures. This is achieved by including a variable $d \times d$ symmetric positive definite matrix norm A_i as an optimising parameter in the objective function J_m.

2.4.6.1 Gustafson-Kessel Algorithm

The A_i norm requires a volume constraint for each ith cluster to limit its growth. This constraint is expressed as $|A_i| = \rho_i$, where $|A_i|$ denotes the determinant of A_i of the ith cluster and ρ_i is its volume. To obtain optimised solutions for the three cluster parameters u_{ik}, $\mathbf{v}_i$ and A_i, we begin by defining an objective function

$$J_m(U,V) = \sum_{k=1}^{N} \sum_{i=1}^{c} u_{ik}^m d_{ik}^2 \tag{2.4.23}$$

where

$$d_{ik}^2 = (\mathbf{x}_k - \mathbf{v}_i)^T A_i (\mathbf{x}_k - \mathbf{v}_i) \tag{2.4.24}$$

with identical interpretations of $\mathbf{x}_k$ and $\mathbf{v}_i$ to FCM. The two constraints are:

$$\text{(i) membership: } u_{ik} \in [0,1], \forall i,k \quad 0 < \sum_{k=1}^{N} u_{ik} < N, \forall i \quad \text{and} \quad \sum_{i=1}^{c} u_{ik} = 1, \forall k \tag{2.4.25}$$

$$\text{(ii) } A_i \text{ matrix: } |A_i| = \rho_i \tag{2.4.26}$$

We may minimise J_m for each of the cluster parameters u_{ik}, $\mathbf{v}_i$ and A_i, subject to the two constraints, by applying the Lagrange multiplier λ to the membership and γ to the matrix and setting their derivatives to zero. In other words, define

$$J_m(U,V) = \lambda g + \gamma h \tag{2.4.27}$$

where $g = \sum_{i=1}^{c} u_{ik} - 1 = 0$

and $h = \sum_{i=1}^{c} \{|A_i| - \rho_i\} = 0$

Then

$$\frac{\partial J_m}{\partial u} = 0 \Rightarrow u_{ik} = \frac{1}{\sum_{j=1}^{c}\left[\frac{d_{ik}^2}{d_{jk}^2}\right]^{\frac{1}{m-1}}} \tag{2.4.28}$$

$$\frac{\partial J_m}{\partial v} = 0 \Rightarrow \mathbf{v}_i = \frac{\sum_{k=1}^{N} u_{ik}^m \mathbf{x}_k}{\sum_{k=1}^{N} u_{ik}^m} \tag{2.4.29}$$

$$\frac{\partial J_m}{\partial A} = \sum_{k=1}^{N} u_{ik}^m (\mathbf{x}_k - \mathbf{v}_i)(\mathbf{x}_k - \mathbf{v}_i)^T + \gamma |A_i| A_i^{-1} = 0 \tag{2.4.30}$$

where in (2.4.30) the identities $\frac{\partial (X^T A X)}{\partial A} = XX^T$ and $\frac{\partial |A|}{\partial A} = |A| A^{-1}$ are used.

Solving for γ yields

$$A_i = F_i^{-1}\left[\rho_i |F_i|\right]^{1/d} \tag{2.4.31}$$

where d is the dimensionality of the data vector and

$$F_i = \frac{\sum_{k=1}^{N} u_{ik}^m (\mathbf{x}_k - \mathbf{v}_i)(\mathbf{x}_k - \mathbf{v}_i)^T}{\sum_{k=1}^{N} u_{ik}^m} \tag{2.4.32}$$

is a positive definite $d \times d$ matrix. Equation (2.4.32) is called the *fuzzy covariance matrix* because of the fuzzy memberships and the covariance terms in the numerator. Putting (2.4.31) and (2.4.32) into (2.4.24), yield a metric given by

$$d_{ik}^2 = \rho_i |F_i|^{1/d} (\mathbf{x}_k - \mathbf{v}_i)^T F_i^{-1} (\mathbf{x}_k - \mathbf{v}_i) \tag{2.4.33}$$

In their paper, Gustafson and Kessel [1979] demonstrated the algorithm's successful detection of 2 clusters consisting of 20 points in the shape of a cross, intersecting at the cluster centroids. Although Gustafson and Kessel gave no criteria for establishing the value of ρ_i, others [Gath and Geva, 1989; Krishnapuram, 1994] have obtained satisfactory results with $\rho_i = 1$. The iterative optimisation of the cluster parameters proceed along similar steps as the FCM, and involves initial estimate of F_i in (2.4.32) with update of memberships from (2.4.28) and the prototypes from (2.4.29).

Proceeding from statistical assumptions, Gustafson and Kessel obtained for the maximum likelihood estimation

$$\Sigma_i^{-1} = \frac{\sum_k p(\mathbf{x}_k|w_i)(\mathbf{x}_k - \boldsymbol{\mu}_i)(\mathbf{x}_k - \boldsymbol{\mu}_i)^T}{\sum_k p(\mathbf{x}_k|w_i)} \tag{2.4.34}$$

For the special case where $\mathbf{x}_k$ is conditionally Gaussian distributed

$$\log p(\mathbf{x}_k|w_i) = -\tfrac{d}{2}\log 2\pi + \tfrac{1}{2}\log|\Sigma_i^{-1}| - \tfrac{1}{2}(\mathbf{x}_k - \boldsymbol{\mu}_i)^T \Sigma_i^{-1}(\mathbf{x}_k - \boldsymbol{\mu}_i) \tag{2.4.35}$$

where $|\Sigma_i^{-1}|$ is assumed non-singular and

$$\boldsymbol{\mu}_i = \frac{\sum_k^N p(\mathbf{x}_k|w_i)\mathbf{x}_k}{\sum_k^N p(\mathbf{x}_k|w_i)} \tag{2.4.36}$$

Comparing (2.4.36) with (2.4.29), and (2.4.34) with (2.4.32) it can be seen that $p(\mathbf{x}_k|w_i) = u_{ik}^m$. This implies that the fuzzy covariance matrix is analogous to the statistical maximum likelihood estimation of mixture densities. Consequently, the Bayes decision rule for the assignment of $\mathbf{x}_j$ to class w_i if $p(\mathbf{x}_j|w_i) > p(\mathbf{x}_j|w_k)$ for all $i \neq k$ has the interpretation of assigning $\mathbf{x}_j$ to the ith cluster if $u_{ij} > u_{kj}$. This interesting result indicates that the partitioning by fuzzy memberships is analogous to Bayes rule for optimal classification.

2.4.6.2 Gath-Geva Algorithm

Gath and Geva [1989] extended the Gustafson-Kessel's algorithm to perform unsupervised optimal clustering based on two cluster validity measures called the *fuzzy hypervolume*, F_{HV} and the *partitional density* P_D. These measures partition X optimally in the sense of the following three criteria:

1. Clear separation between clusters.
2. Minimal volume of clusters.
3. Maximal number of data points.

Gath and Geva employed a two layer clustering strategy, in which the first layer corresponds to FCM. The prototypes identified by FCM is used in the second layer to obtain maximal partitions from F_{HV} and P_D fuzzy measures. All the equations used are similar to Gustafson and Kessel's equations, except for the distance measure, reformulated as an exponential distance measure based on maximum likelihood estimation

$$d_{ik} = \frac{|F_i|^{1/2}}{\frac{1}{N}\sum_{k=1}^{N} u_{ik}} \exp(\mathbf{x}_k - \mathbf{v}_i)^T F_i^{-1}(\mathbf{x}_k - \mathbf{v}_i) \tag{2.4.37}$$

Note that (2.4.37) of Gath and Geva is similar to (2.4.33) of Gustafson and Kessel, except for the exponential form of the distance measure. Equation (2.4.37) assumes $m = 2$ and $d = 2$. The fuzzy hypervolume is defined as

$$F_{HV} = \sum_{i=1}^{c} |F_i|^{\frac{1}{2}} \tag{2.4.38}$$

and the partitional density is calculated from

$$P_D = \frac{S}{F_{HV}} \tag{2.4.39}$$

where

$$S = \sum_{i=1}^{c}\sum_{k=1}^{N} u_{ik} \qquad \forall \mathbf{x}_k \in \left\{\mathbf{x}_k : (\mathbf{x}_k - \mathbf{v}_i) F_i^{-1} (\mathbf{x}_k - \mathbf{v}_i) < 1\right\} \tag{2.4.40}$$

In (2.4.40), $\mathbf{x}_k$ is constrained to less than 1 unit distance from the hypervolume. Note that both the Gath-Geva's algorithm and Gustafson-Kessel's algorithm use the same membership function (2.4.28). Therefore, both may be generalised by the PFCM.

2.4.7 Possibilistic Memberships

In this section, a class of fuzzy clustering algorithm with possibilistic memberships is to be defined by (2.4.41) and (2.4.42). This type of algorithm usually depends on a first layer algorithm to facilitate iteratively optimised solutions for the cluster parameters u_{ik} and $\mathbf{v}_i$.

2.4.7.1 Possibilistic *c*-Means (PCM)

The possibilistic *c*-Means (PCM) was proposed by Krishnapuram and Keller [1993a] to obtain clusters that correspond more closely to the intuitive concept of *typicality* or *compatibility*. PCM reformulates FCM membership as a function of the distance of a point from its prototype. This is achieved by relaxing the membership constraint of (2.4.10) to

$$u_{ik} \in [0,1] \quad \forall i,k \quad 0 < \sum_{k=1}^{N} u_{ik} \le N \quad \forall i \quad \text{and} \quad \max_i \{u_{ik}\} > 0 \quad \forall k \tag{2.4.41}$$

The objective function satisfying this requirement is

$$J_m(U,V) = \sum_{i=1}^{c}\sum_{k=1}^{N} u_{ik}^m d_{ik}^2 + \sum_{i=1}^{c} \eta_i \sum_{k=1}^{N} (1 - u_{ik})^m \tag{2.4.42}$$

where η_i is a factor of the intra-cluster distance of the ith cluster. The first term of (2.4.42) represents the FCM objective function and therefore inherits similar FCM properties for metric d_{ik}. The second term is new and means that J_m is optimised for minima of $(1 - u_{ik})$, where η_i is a positive constant of the cluster that determines the weighting of the second term relative to the first. In other words, u_{ik} is maximised to make the second term as small as possible. Because there are no specific membership constraints to satisfy, other than (2.4.41), the objective function may be minimised with respect to u_{ik} to give

$$u_{ik} = \frac{1}{1+\left(\frac{d_{ik}^2}{\eta_i}\right)^{\frac{1}{m-1}}} \tag{2.4.43}$$

which represents *possibilistic* memberships in contrast to the *probabilistic* memberships of (2.4.11). The point memberships of $\mathbf{x}_k$ in (2.4.43) depend only on a single cluster, unlike FCM's memberships of (2.4.11.). Krishnapuram and Keller proposed two equations for determining η_i. The first equation

$$\eta_i = \frac{\sum_{k=1}^{N} u_{ik}^m d_{ik}^2}{\sum_{k=1}^{N} u_{ik}^m} \tag{2.4.44}$$

makes η_i an average of the fuzzy intra-cluster distance of the ith cluster. This form is used initially when the cluster prototypes are not known. If the prototypes are estimated from FCM, a more accurate form of the equation is given by

$$\eta_i = \frac{\sum_{k=1}^{N} d_{ik}^2 F_{ik}}{\sum_{k=1}^{N} F_{ik}} \tag{2.4.45}$$

$$F_{ik} = \begin{cases} 1 & \text{if} \quad u_{ik} > \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

where α_t is the alphacut, a threshold on the membership u_{ik}. Practical problems with η_i are not satisfactorily solved. The effect of η_i on clustering performance appears to be difficult to control and good results are obtained only for low noise situations. Recognising the difficulties of estimating η_i, Krishnapuram in a later paper [1994] proposed an alternative solution for η_i by equating (2.4.43) with

$$u_{ik} = \exp\left[-\tfrac{1}{2}(\mathbf{x}_k - \mathbf{v}_i)F_i^{-1}(\mathbf{x}_k - \mathbf{v}_i)\right] \tag{2.4.46}$$

Note that the membership function of (2.4.46) is very similar to Gath-Geva's metric in (2.4.37) except for a constant term. The negative sign is present in the exponent of (2.4.46) because membership in possibilistic functions is inversely related to the distance measure given by (2.4.37).

2.4.7.2 Enhanced Possibilistic *c*-Means (EPCM)

Bezdek and Im's Enhanced Possibilistic *c*-Means (EPCM) bears a close structural relation to the PCM. We present the objective function corresponding to (2.4.46) (omitted in [Krishnapuram and Keller, 1994]), but with a slight difference which we shall see in a moment. Assume an objective function of the form

$$J_m(U,V) = \sum_{i=1}^{c}\sum_{k=1}^{N} u_{ik}^m e^{d_{ik}/\eta_i} + \sum_{i=1}^{c}\sum_{k=1}^{N}\left[1 - \left(\frac{m}{m-1}\right) u_{ik}^{m-1}\right] \tag{2.4.47}$$

for $m > 1$, and for generality assume $d_{ik} = \|\mathbf{x}_k - \mathbf{v}_i\|_A$ is a norm with a $d \times d$ positive definite matrix A. The second term of (2.4.47) admits a similar interpretation as in (2.4.42) since m is a constant except without the intra-cluster scaling factor. Then u_{ik} may be a local minima of J_m with the expression

$$u_{ik} = \frac{1}{\exp\left(d_{ik}/\eta_i\right)} \tag{2.4.48}$$

It is apparent that (2.4.48) equals (2.4.46) if we include a fuzzy covariance matrix $\frac{1}{2}F_i^{-1}$ in d_{ik} and assume $\eta_i = 1$. The form of (2.4.48) has a possibilistic distribution. One of the advantage in expressing the membership function in the form of (2.4.48) is the avoidance of the singularity problem present in FCM. With possibilistic membership, $d_{ik} = 0$ results in unit membership. The η_i term controls the cluster bandwidth. It is also possible to use η_i as a constant, independent of the cluster variable i. If the clusters are known a priori, η_i can be estimated approximately from the average of the cluster mean radius; otherwise it is necessary to guess an initial value for η_i. The prototype solution corresponding to local minima of J_m is given by

$$\mathbf{v}_i = \frac{\sum_{k=1}^{N} u_{ik}^m \mathbf{x}_k}{\sum_{k=1}^{N} u_{ik}^m} \tag{2.4.49}$$

which is identical to FCM after replacing the fuzzifier exponent $(m - 1)$ with m. The expression for the intra-cluster distance is given by

$$\eta_i = \frac{\sum_{k=1}^{N} d_{ik} F_{ik}}{\sum_{k=1}^{N} F_{ik}} \tag{2.4.50}$$

$$F_{ik} = \begin{cases} 1 & \text{if} \quad u_{ik} > \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

2.4.7.3 Comparing FCM, PCM and EPCM Clustering Performance

One misconception of possibilistic clustering (see [Krishnapuam, 1994]) is the attribution of centroid clustering tendency from a possibilistic distribution. As will be demonstrate shortly, this property is a function of the exponent parameters of the clustering criterion, not its possibilistic distribution. Undoubtedly, one cogent reason for this view is because the particular choice of $p = 2$ does not yield good prototypes for the FCM algorithm.

Table 2.9 shows that there exist values of m and p in (2.4.19) which produce centroid clustering (prototypes shown shaded) from both FCM and PFCM, for the data sets of Kaufman and Rousseeuw (Fig. 2.11), and Krishnapuram and Keller (Fig. 2.12). Under a normal value of $p = 2$, FCM does not cluster well to centroid. However, a smaller $p = 1.2$ gives good prototypes at centroids. The table shows that PCM clusters poorly at the recommended values of α_t ($0.1 \le \alpha_t \le 0.5$, from [Krishnapuram and Keller, 1993]). On the data sets of Figs. 2.11 and 2.12, PCM gives a less satisfactory result compared to FCM at normal cluster parameter settings ($m = p = 2$). This is attributed to the effect of α_t on the points in N_α. In other words, a centred prototype is obtained from good points in N_α such that $\alpha_t > u_{noise}$ (u_{noise} is a noise point, eg. point *6* or *13* in Fig. 2.11). All three algorithms clustered at local centroids for the noise free data of Table 2.5.

Tables 2.10 and 2.11 show the cluster prototype results using the recommended procedure suggested in [Krishnapuram and Keller, 1993]. For both sets of data, especially that of Kaufman and Rousseeuw's data set, PCM showed a high degree of sensitivity to the initial FCM membership values used to the estimate η_i. In the case of Krishnapuram and Keller's well-separated and compact cluster data (with noise) of Fig. 2.12, the procedure yielded accurately centred prototypes only for $\alpha_t \ge 0.4$, as indicated in Table 2.11. For the less compact three cluster data of Kaufman and Rousseeuw (Fig. 2.11), PCM's clustering

performance deteriorated significantly, finding good cluster prototypes only at $\alpha_t = 0.3$, as shown in Table 2.10. This is due to PCM's sensitivity to alphacut α_t.

Algorithm	Parameters	η_i	Prototypes	Data
FCM	m=2, p=2		2.39, 9.19 7, 2.13 13.31, 9.09	KR
	m=1.8, p=1.2		2.03, 9.01 7, 2 13.39, 9.05	
	m=2, p=2		59.98, 150 140.02, 150	KK (no noise)
	m=2, p=1		60, 150 140, 150	KK (with noise)
PFCM1	α=7.5, m=15, p=1.2		2.01,9 7,2 13.39,9.05	KR
	α=1, m=2, p=2		59.98,150 140,150	KK (no noise)
	α=1, m=2, p=2		62.78,155.48 137.23,155.48	KK (with noise)
	α=1, m=2, p=2		60,150 140,150	
PFCM2	m=0.1, p=1.1		2,9 7,1.99 13.51,9	KR
	α=1, m=2, p=2		59.98,150 140.02,150	KK (no noise)
	α=1, m=2, p=2		62.78,155.48 137.23,155.48	KK (with noise)
	α=.075, m=.1, p=1.1		59.98,150 140.02,150	
PCM	α_t=1, m=2	10.78 3.59 14.73	2.15,8.91 7,2.03 12.9,8.7	KR
	m=2 (fixed η)	100	60.07,150 139.93,150	KK (no noise)
	α_t=0.2, m=2	1.96E+3 1.96E+3	67.44,151.3 132.56,151.3	KK (with noise)

Table 2.9 Summary of clustering characteristics. The true centroids of clusters are at (2, 9), (7, 2) and (13.5, 9) for Fig. 2.11 (Kaufman and Rousseeuw's data set), and (60, 150) and (140, 150) for Fig. 2.12 (Krishnapuram and Keller's data set). Symbols KR = Kaufman and Rousseeuw, and KK = Krishnapuram and Keller. Prototypes centred at centroids are shown shaded.

Method	α_t	η_i (at 25 iterations)	Prototypes
1. FCM (1-8 iter)	0.1	10.78,16.06,14.73	(7.09,3.91),(7.09,3.99),(7.16,4.50)
2. Estimate η_i at 8th iteration	0.2	10.78,3.59,14.73	(2.15,8.91),(7,2.03),(12.9,8.7)
3. PCM (9-25 iter)	0.3	10.78,3.59,10.52	(2.15,8.91),(7,2.03),(13.4,8.7)
4. Re-estimate η_i at 25th iteration	0.4	8.23,3.59,1.63	(2.02,8.99),(7,2.03),(13.4,8.44)
5. PCM (26-40 iter)	0.5	0.985,1.35,1.63	(2.01,8.99),(7,2.03),(13.42,8.36)

Table 2.10 PCM clustering result for Kaufman and Rousseeuw's data set (Fig. 2.11). True centroids are located at (2, 9), (7, 2) and (13.5, 9). Cluster parameter $m = 2$.

Method	α_t	η_i (at 25 iterations)	Prototypes
1. FCM (1-8 iter)	0.1	3.47E3,3.47E3	(81.7,152.98),(118.3,152.98)
2. Estimate η_i at 8th iteration	0.2	2.7E3,2.7E3	(73.08,152.08),(126.92,152.08)
3. PCM (9-25 iter)	0.3	1.17E3,1.17E3	(63.18,150.59),(136.82,10.59)
4. Re-estimate η_i at 25th iteration	0.4	407.18,407.18	(60.6,150.11),(139.4,150.11)
5. PCM (26-40 iter)	0.5	20,99.93	(60.01,150),(139.91,150)
	0.6	20,20	(60.01,150),(139.99,150)
	0.7	20,20	(60.01,150),(139.99,150)
	0.9	20,20	(60,140),(140,150)

Table 2.11 PCM clustering result for Krishnapuram and Keller's data set with noise (Fig. 2.12). True centroids are located at (60, 150) and (140, 150). Cluster parameter $m = 2$.

A practical problem with Krishnapuram and Keller's procedure is that it is difficult to use since PCM's η_i is quite sensitive to FCM memberships. Perhaps this is less of a problem with denser cluster points and a larger number of points in the data set. We investigated combinations of high and low m on both data sets, for both PCM and FCM algorithms, but none seemed to produce acceptable and consistent results over the whole or partial range of α_t. Consequently, Bezdek and Im's EPCM algorithm was developed to overcome this problem. Table 2.12 and 2.13 show that good cluster prototype results were obtained from EPCM over a wide range of α_t.

Method	α_t	η_i (at 8 iterations)	Prototypes
1. FCM (1-8 iter)	0.1	2.45,2.42,2.75	(2.07,8.96),(7,2.05),(13.4,8.94)
2. Estimate η_i at 8th iteration	0.2	2.45,1.51,2.75	(2.07,8.96),(7,2.01),(13.4,8.94)
3. EPCM (9-33iter)	0.3	2.45,1.51,2.27	(2.07,8.96),(7,2.01),(13.46,8.99)
	0.4	2.45,1.18,1.39	(2.07,8.96),(7,2),(13.5,9.08)
	0.5	2.13,1.18,1.39	(2.03,8.98),(7,2),(13.5,9.08
	0.6-0.9	2.13,1.18,1.39	(2,9),(7,2),(13.5,9.08)

Table 2.12 EPCM clustering result for Kaufman and Rousseeuw's data set (Fig. 2.11). True centroids are located at (2, 9), (7, 2) and (13.5, 9). Cluster parameters are $m = 2$, $p = 2.6$ for FCM and $m = 2$ for EPCM.

Method	α_t	η_i (at 8 iterations)	Prototypes
1. FCM (1-8 iter) 2. Estimate η_i at 8th iteration 3. EPCM (9-33iter)	0.1 to 0.9	5.71,5.71	(60,150),(140,150)

Table 2.13 EPCM clustering result for Krishnapuram and Keller's data set with noise (Fig. 2.12). True centroids are located at (60, 150) and (140, 150). Cluster parameters are $m = 1.3$ and $p = 2$ for FCM and $m = 5$ for EPCM.

The EPCM algorithm yielded satisfactory cluster prototypes over the entire range of $0.1 \leq \alpha_t \leq 0.9$. This result requires a judicious choice of the right combinations of cluster parameters m and p for FCM and m for EPCM. Two variations on the cluster parameters give an acceptable result: (i) normal m for both FCM and EPCM, and high p for FCM (Table 2.12) and (ii) normal p with low m for FCM and high m for EPCM (Table 2.13). Other combinations are possible, but have not been investigated. In general, a satisfactory procedure is to proceed by trial an error until a consistent cluster result is obtained, typically by varying a single cluster parameter at a time. In this case, it is easier to work with initial crisp partitions from FCM ($1.1 \leq m \leq 1.5$) and to vary m in EPCM. We have only considered EPCM in conjunction with FCM because of the popularity of FCM. The possibility of using PFCM is good and should improve the cluster result.

With regard to the relationship of fuzzy algorithms to noise, it cannot be asserted that any of the PCM, EPCM or PFCM has an advantage over FCM, for two reasons. A possibilistic function is not a sufficient criterion for classification purposes since neighbouring clusters are also significant. Moreover, a possibilistic function has no intrinsic attribute that makes it less sensitive to noise compared to FCM. According to the results of the Tables 2.9 and 2.10, all three algorithms suffer from unsatisfactory cluster prototypes,

and for the reason given. Noise insensitivity is to a certain degree, obtained from the statistical process of averaging and assisted by the choice of the error criterion. We have demonstrated in Table 2.9 that suitable values of m and p produce centred prototypes from FCM. Possibilistic algorithms like PCM, EPCM and PFCM on the one hand, and FCM algorithm and its derivatives on the other, are best considered to be alternative methods of clustering. The good clustering results from PFCM2 and the similarity of clustering performance to PFCM1 suggest some freedom in applying the optimising criterion to solve for prototypes $\mathbf{v}_i$. As noted previously, this produces a more efficient algorithm.

For the development of our experimental algorithms discussed in Section 3, we will use the EPCM membership equation, for the reasons of clustering effectiveness and efficiency (other FCM equations are not excluded).

2.4.8 Parametrized Prototypes

In theory, the distance measure of the fuzzy covariance matrix of Gustafson-Kessel (2.4.33) and of the Gath-Geva (2.4.37), or the Possibilistic variant of the Gustafson-Kessel's memberships (2.4.46) will detect a variety of elliptic cluster structures. This mode of fuzzy clustering performs an image processing operation called segmentation. The same result can of course be achieved by other methods such as the Generalised Hough Transform. However, the fuzzy approach solves the segmentation problem more efficiently and elegantly. More significantly, the fuzzy objective function approach provides a systematic procedure to optimise solutions for cluster parameters, in the sense of minimising the clustering criteria.

In this section, we will examine more elaborate fuzzy methods to parametrize the prototypes, ie. defining the prototype parameters to enable sophisticated clustering. We begin with the Man and Gath's ring shape detection fuzzy clustering algorithm [1994] and conclude with an extension of Gath and Hoory's algorithm to detect elliptic outlines [1995]. In both cases, we present our alternative solutions to demonstrate the procedure for solving this class of clustering problems.

2.4.8.1 Fuzzy *K*-Rings [Man and Gath, 1994]

For the ring parameters represented in Fig. 2.13, let the objective function be

$$J_m = \sum_{k=1}^{N}\sum_{i=1}^{c} u_{ik}^m D_{ik}^2 \tag{2.4.51}$$

The distance measure is defined as

$$D_{ik}^2 = (\mathbf{x}_k - \mathbf{V}_i)^T (\mathbf{x}_k - \mathbf{V}_i) \tag{2.4.52}$$

and

$$\mathbf{x}_k = \begin{bmatrix} x_{1,k} \\ x_{2,k} \end{bmatrix} \tag{2.4.53}$$

$$\mathbf{V}_i = \begin{bmatrix} v_{1,i} + r_i \cos\theta_i \\ v_{2,i} + r_i \sin\theta_i \end{bmatrix} \tag{2.4.54}$$

where

$\mathbf{v}_i = [v_{1,i}, v_{2,i}]^T$ and $\|\mathbf{r}_i\| = r_i$

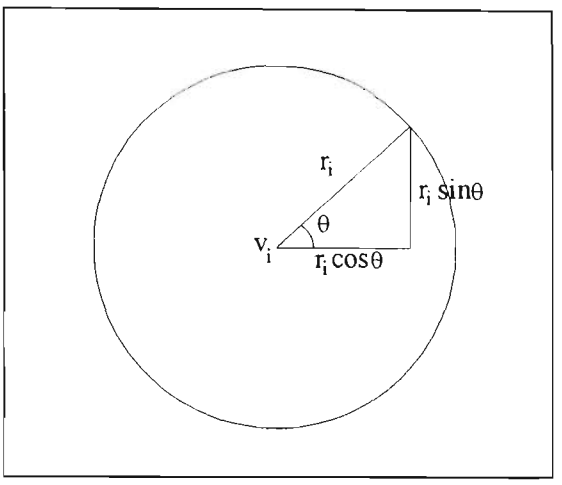


Figure 2.13 A schematic of ring cluster parameters. The cluster centre is at $\mathbf{v}_i$, and the cluster radius is r_i. Horizontal and vertical components of r_i are $r_i \cos\theta_i$ and $r_i \sin\theta_i$ respectively.

The circle centre point is denoted by $\mathbf{v}_i$. The components of data point $\mathbf{x}_k$ are $x_{1,k}$ and $x_{2,k}$. The cluster parameters $\mathbf{v}_i$, $\mathbf{r}_i$ and θ_i are defined in Fig. 2.13. The parametrized prototype equation (2.4.54) includes two cluster parameters $\mathbf{r}_i$ and $\mathbf{v}_i$ for optimisation. Representing the cluster prototypes in this way means that when $d_{ik} = 0$, the condition for local minima of J_m is satisfied. The optimised solutions for u_{ik}, $\mathbf{v}_i$ and $\mathbf{r}_i$ may be found by minimising J_m of (2.4.51) with respect to u_{ik}, $\mathbf{v}_i$ and $\mathbf{r}_i$ respectively. Since u_{ik} of the objective function is identical to FCM, the optimised solution for u_{ik} will be identical to FCM. However, the additional parameters in (2.4.54) are expected to produce a new solution which resembles FCM. Minimising J_m with respect to $\mathbf{v}_i$ gives

$$\frac{\partial J_m}{\partial v} = \sum_{k=1}^{N} u_{ik}^m \frac{\partial (\mathbf{x}_k - \mathbf{V}_i)^T (\mathbf{x}_k - \mathbf{V}_i)}{\partial v_i} = -2 \sum_{k=1}^{N} u_{ik}^m (\mathbf{x}_k - \mathbf{V}_i) = 0 \tag{2.4.55}$$

Substituting (2.4.53) and (2.4.54) into (2.4.55) yield the prototype solutions

$$\mathbf{v}_i = \begin{bmatrix} v_{1,i} \\ v_{2,i} \end{bmatrix} = \frac{\sum_{k=1}^{N} u_{ik}^m \begin{bmatrix} x_{1,k} - r_i \cos\theta_i \\ x_{2,k} - r_i \sin\theta_i \end{bmatrix}}{\sum_{k=1}^{N} u_{ik}^m} \tag{2.4.56}$$

From (2.4.52)

$$D_{ik} = \|\mathbf{x}_k - \mathbf{V}_i\| = \|(\mathbf{x}_k - \mathbf{v}_i) - \mathbf{r}_i)\| = \|d_{ik} - \mathbf{r}_i\|$$

Minimising J_m of (2.4.51) with respect to $\mathbf{r}_i$ results in

$$\mathbf{r}_i = \frac{\sum_{k=1}^{N} u_{ik}^m \|\mathbf{x}_k - \mathbf{v}_i\|}{\sum_{k=1}^{N} u_{ik}^m} \tag{2.4.57}$$

The form of (2.4.56) resembles the FCM solution, translated by a distance corresponding to the components of r_i. The algorithm based on these equations is called the Fuzzy K-Rings (FKR) by their inventors. Since the Man-Gath algorithm has almost the same basic equations as the FCM, except for two additional parameters for the prototypes, it may be iteratively optimised like the FCM algorithm.

2.4.8.2 Fuzzy *K*-Ellipse [Gath and Hoory, 1995]

An algorithm that detects elliptic cluster outlines is called by their inventors, the Fuzzy K-Ellipse (FKE). Like the FKR, the FKE is based on the optimisation of an objective function in which the set of elliptic prototypes is parametrized for optimisation. As shown in Fig. 2.14, each ellipse is uniquely defined by a radius $r_i = 2a_i$ and two foci. The five parameters of the ellipse are represented by the two dimensional foci vectors $\mathbf{v}_i^{(1)}$ and $\mathbf{v}_i^{(2)}$, and the radius r_i. Note the superscript in $\mathbf{v}_i$ is only a label, not a mathematical operator. The two foci are derived from the centre $\mathbf{v}_i$, the focus length f_i and the tilt angle θ_i.

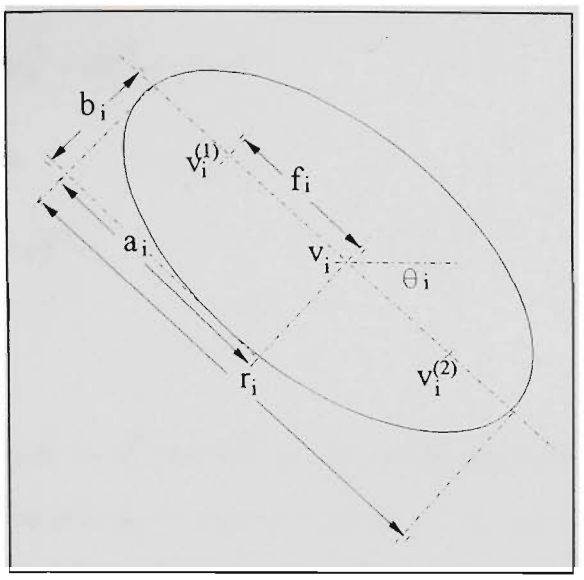


Figure 2.14 A schematic of ellipse cluster parameters. The two foci are located at $\mathbf{v}_i^{(1)}$ and $\mathbf{v}_i^{(2)}$ and $\mathbf{v}_i$ is the centre of the ellipse. The semi-major and semi-minor axes are represented by a_i and b_i respectively. θ_i is the tilt angle.

For the ellipse centred at $\mathbf{v}_i$, let

$$\mathbf{v}_i^{(1)} = \begin{bmatrix} v_{1,i}^{(1)} - f_i \cos\theta_i \\ v_{2,i}^{(1)} - f_i \sin\theta_i \end{bmatrix} \tag{2.4.58}$$

$$\mathbf{v}_i^{(2)} = \begin{bmatrix} v_{1,i}^{(2)} + f_i \cos\theta_i \\ v_{2,i}^{(2)} + f_i \sin\theta_i \end{bmatrix} \tag{2.4.59}$$

where

$$f_i = \sqrt{a_i^2 - b_i^2} \tag{2.4.60}$$

for $a_i \geq b_i$ and a_i is the semi-major axis. Define

$$D_{ik} = |d_{ik}^{(1)} + d_{ik}^{(2)} - r_i| \tag{2.4.61}$$

where

$$d_{ik}^{(1)} = \|\mathbf{x}_k - \mathbf{v}_i^{(1)}\| \tag{2.4.62}$$

$$d_{ik}^{(2)} = \|\mathbf{x}_k - \mathbf{v}_i^{(2)}\| \tag{2.4.63}$$

Note that (2.4.61) is the geometric definition of an ellipse. Define the objective function as

$$J_m(U, V^{(1)}, V^{(2)}, R) = \sum_{k=1}^{N}\sum_{i=1}^{c} u_{ik}^m D_{ik}^2 \tag{2.4.64}$$

where U represents the $c \times N$ membership matrix, V is the c-tuples of prototypes and R the c-tuples of corresponding radii. Minimising J_m with respect to u_{ik} gives the same result as FCM's membership (2.4.11), with the metric D_{ik}. Minimising J_m with respect to r_i gives

$$\frac{\partial J_m}{\partial r} = -2\sum_{k=1}^{N} u_{ik}^m (d_{ik}^{(1)} + d_{ik}^{(2)} - r_i) = 0$$

which yields the solution

$$r_i = \frac{\sum_{k=1}^{N} u_{ik}^m (d_{ik}^{(1)} + d_{ik}^{(2)})}{\sum_{k=1}^{N} u_{ik}^m} \tag{2.4.65}$$

Minimising J_m with respect to $\mathbf{v}_i^{(1)}$and $\mathbf{v}_i^{(2)}$ by direct differentiation involves lengthy derivations. However, a simpler way to solve it is given by the solutions of the Man-Gath's algorithm. In the Man-Gath's algorithm, we noticed that (2.4.53) and (2.4.54) produced a solution given by (2.4.56). Since the form of the equations are identical in both cases, we may deduce a solution by deriving the equivalent expressions to (2.4.53) and (2.4.54). Let

$$\mathbf{x}_k = \begin{bmatrix} x_{1,k} \\ x_{2,k} \end{bmatrix}$$

From the geometric property of an ellipse,

$$\|\mathbf{x}_k - \mathbf{v}_i^{(1)}\| + \|\mathbf{x}_k - \mathbf{v}_i^{(2)}\| = 2a_i = r_i \tag{2.4.66}$$

Solutions exist when $\mathbf{x}_k = \mathbf{V}_i^{(1)}$ and when $\mathbf{x}_k = \mathbf{V}_i^{(2)}$. Considering the first case, this occurs at

$$\mathbf{V}_i^{(1)} = \begin{bmatrix} v_{1,i}^{(1)} + \|\mathbf{x}_k - \mathbf{v}_i^{(1)}\| \cos\theta_i \\ v_{2,i}^{(1)} + \|\mathbf{x}_k - \mathbf{v}_i^{(1)}\| \sin\theta_i \end{bmatrix} \tag{2.4.67}$$

where

$$\mathbf{v}_i^{(1)} = \begin{bmatrix} v_{1,i}^{(1)} \\ v_{2,i}^{(2)} \end{bmatrix} \tag{2.4.68}$$

$$\mathbf{v}_i^{(2)} = \begin{bmatrix} v_{1,i}^{(2)} \\ v_{2,i}^{(2)} \end{bmatrix} \tag{2.4.69}$$

Replacing the terms in (2.4.66) with (2.4.62) and (2.4.63), and substituting the result in (2.4.67) gives

$$\mathbf{V}_i^{(1)} = \begin{bmatrix} v_{1,i}^{(1)} + (r_i - d_{ik}^{(2)}) \cos\theta_i \\ v_{2,i}^{(1)} + (r_i - d_{ik}^{(2)}) \sin\theta_i \end{bmatrix} \tag{2.4.70}$$

By a similar reasoning, we obtain for the second case

$$\mathbf{V}_i^{(2)} = \begin{bmatrix} v_{1,i}^{(2)} + (r_i - d_{ik}^{(1)}) \cos\theta_i \\ v_{2,i}^{(2)} + (r_i - d_{ik}^{(1)}) \sin\theta_i \end{bmatrix} \tag{2.4.71}$$

Comparing (2.4.70) and (2.4.71) with (2.4.56), we deduce the prototype solutions

$$\mathbf{v}_i^{(1)} = \frac{\sum_{k=1}^{N} u_{ik}^m \begin{bmatrix} x_{1,k} - (r_i - d_{ik}^{(2)})\cos\theta_i \\ x_{2,k} - (r_i - d_{ik}^{(2)})\sin\theta_i \end{bmatrix}}{\sum_{k=1}^{N} u_{ik}^m} \tag{2.4.72}$$

$$\mathbf{v}_i^{(2)} = \frac{\sum_{k=1}^{N} u_{ik}^m \begin{bmatrix} x_{1,k} - (r_i - d_{ik}^{(1)})\cos\theta_i \\ x_{2,k} - (r_i - d_{ik}^{(1)})\sin\theta_i \end{bmatrix}}{\sum_{k=1}^{N} u_{ik}^m} \tag{2.4.73}$$

for all i and $1 \le i \le c$. To solve for the cluster parameters a_i, b_i and θ_i, first diagonalise the fuzzy covariance positive definite 2×2 matrix of the ith cluster given by

$$F_i = \frac{\sum_{k=1}^{N} u_{ik}^m (\mathbf{x}_k - \mathbf{v}_i)(\mathbf{x}_k - \mathbf{v}_i)^T}{\sum_{k=1}^{N} u_{ik}^m} \tag{2.4.74}$$

Diagonalising F_i gives

$$E_i^T F_i E_i = D_i$$

which leads to

$$F_i = E_i D_i E_i^T \tag{2.4.75}$$

where $E_i^T = E_i^{-1}$ is an orthogonal eigenvector matrix and D_i is a diagonal eigenvalue matrix. These matrices of the ith cluster have the cluster parameters

$$E_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \\ \sin\theta_i & \cos\theta_i \end{bmatrix} \tag{2.4.76}$$

$$D_i = \begin{bmatrix} 1/a_i^2 & 0 \\ 0 & 1/b_i^2 \end{bmatrix} \tag{2.4.77}$$

where θ_i is the tilt angle of the ellipse, and a_i and b_i are the semi-major and semi-minor axis of the ellipse depicted in Fig. 2.14. The column vectors of E_i are the eigenvectors of F_i and the diagonal elements of D_i are the eigenvalues of F_i. The eigenvalues of F_i are also the coefficients of the principal components of the transformed ellipse, rotated by θ_i and centred at $\mathbf{v}_i$.

The FKE algorithm [Gath and Hoory, 1995].

Step 1. Run the FKR with two clusters, to provide initial estimates for the 2 foci.

Step 2. Fix c, the number of clusters.

Step 3. Set initial U_0 and foci matrices $V^{(1)}$ and $V^{(2)}$.

(a) For exocentric case, start from any partition and run FKR for about 10 iterations.

(b) For concentric case, fix the cluster centres such that they are distributed in the vicinity of the centre.

(c) Run the FKR for 10 iterations.

(d) Calculate the c fuzzy covariances from (2.4.74), their eigenvectors and eigenvalues.

(e) Split each cluster centre into two foci using (2.4.76), (2.4.77), (2.4.58) and (2.4.59).

For each iteration,

Step 4. Update the radii r_i from (2.4.65), for $i = 1,...,c$.

Step 5. Update the foci $\mathbf{v}_i^{(1)}$ and $\mathbf{v}_i^{(2)}$ from (2.4.72) and (2.4.73).

Step 6. Update memberships u_{ik} from (2.4.11) with metric D_{ik}.

Step 7. If $|U_t - U_{t-1}| < \varepsilon$, Stop, else go to Step 4.

Note: ε is a small value to control the stopping point and t is an iteration index.

2.4.9 Fuzzy Partition Space

This section ties together some observations on fuzzy partitions and relationships between fuzzy and crisp clustering. Conventional clustering methods assume an object belongs uniquely to a single class. In practice, this is quite an unrealistic class assignment nor does it best represent the status of the data. The concept of fuzzy subsets offers a more general classification, one that can be tailored to the nature of the data.

Now we review the fuzzy partition space mentioned earlier more formally. Let c be an integer in the range of $1 < c < N$ ($c = N$ is a trivial partition and $c = 1$ is an invalid case) and let $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$ denote a set of N discrete unlabelled feature vectors in real d feature space R^d, where d is the dimensionality of the feature vector. Given X, we say that c fuzzy subsets $\{u_i : X \to [0,1]\}$ are a fuzzy c-partition of X for the case where the cN values of $\{u_{ik} = u_i(\mathbf{x}_k),\ 1 \le k \le N,\ 1 \le i \le c\}$ satisfy the following three conditions:

$$u_{ik} \in [0,1] \quad \forall i,k, \quad 0 < \sum_{k=1}^{N} u_{ik} < N \quad \forall i \quad \text{and} \quad \sum_{i=1}^{c} u_{ik} = 1 \quad \forall k \tag{2.4.78}$$

where u_{ik} is the membership of $\mathbf{x}_k$ in the ith partitioned fuzzy subset (cluster). Each set of $c \times N$ values of (2.4.78) can be organised as a real valued $c \times N$ matrix $U = [u_{ik}]$. The set

of all such matrices is the non-degenerate fuzzy c-partitions (ie. no identical zeros in any row or column of U) of X:

$$M_{fcN} = \{U \text{ in } R^{cN} \mid u_{ik} \text{ satisfying condition (2.4.78), for all } i \text{ and } k)\}$$

In the crisp clustering case (ie. when $m \rightarrow 1^+$, for FCM, or $\alpha \rightarrow m$, for PFCM), all the u_{ik} are either 1 or 0, giving a subset of hard c-partitions of X:

$$M_{cN} = \{U \text{ in } M_{fcN} \mid u_{ik} = 0 \text{ or } 1, \text{ for all } i \text{ and } k)\}$$

It is seen that M_{cN} is a special case of M_{fcN} but not vice versa. M_{fcN} is a more realistic physical model than M_{cN} because M_{fcN} provides a richer means for manipulating data that have ambiguous structures than does M_{cN}. Mathematically M_{fcN}, which is the convex hull of M_{cn}, provides a more tractable and useable set than M_{cN}. It is helpful to be reminded of Bezdek's emphasis that the entries of the fuzzy matrix U are not probabilities, but similarities of object vectors to class paradigms. Furthermore, there are statistical clustering algorithms that produce solutions in M_{fcN} [Duda and Hart, 1973] and fuzzy algorithms that produce analogous statistical solutions [Gustafson and Kessel, 1979]. Finally, it is possible to relax the membership constraint of (2.4.10) so that the sum of the memberships is not required to be unity (as in (2.4.16) and (2.4.41)). We have seen examples of this with the possibilistic variety of algorithms in Section 2.4.7. The relaxation of the membership constraint, according to Bezdek [Bezdek and Pal, 1992, p.15] "is a natural and physically appealing extension of M_{fcN} to an even larger solution space for clustering".

Remarks

The procedure for the Gath-Hoory's algorithm is representative of the latest generation of fuzzy clustering algorithms. It is presented to illustrate some of the "mechanics" of using a fuzzy clustering algorithm. Unfortunately, not many fuzzy clustering algorithms are straight forward to use, the only exception being FCM. The number of cluster parameters to be found can have a significant impact on the complexity of the algorithm and the computation efforts. Moreover, having obtained the clusters additional processing is often necessary to establish cluster validity. These reasons justify a need to develop simpler forms of the fuzzy algorithm. The investigation of alternative solutions with the CPCM based algorithms in following chapters of this thesis, represents an attempt to overcome some of these problems. This approach does not require a complete abandonment of their fuzzy analytical counterparts, but allows useful attributes to be adapted within the CPCM framework, to operate in a manner that is almost analogous to fuzzy clustering algorithms.

Excellent seminal papers on fuzzy models for pattern recognition are collected in a comprehensive volume, edited by Bezdek and Pal [1992]. The following references provide a helpful complement to this topic [Kandel, 1982; Kosko, 1992; Pal, 1986]. Exceptionally informative and well illustrated guide to the subject of fuzzy clustering are the classic tutorials given by Bezdek in Australia [Bezdek, 1995a and 1995b]. For an interesting personal view, by an authoritative advocate and practitioner of fuzzy thinking, see [Kosko, 1993].

Up to date fuzzy clustering and pattern recognition topics are to be found in the following conference proceedings:

- IEEE International Conference on Neural Networks.
- IEEE International Conference on Neural Networks and Signal Processing.
- IEEE International Conference on Acoustics, Speech and Signal Processing.
- IEEE International Conference on Fuzzy Systems.
- IEEE Region Ten Conference on Digital Signal Processing Applications.
- Australia New Zealand Intelligent Information Processing Conference.
- Conference on Digital Image Computing: Techniques and Applications (International Association for Pattern Recognition Inc. and Australian Pattern Recognition Society).
- International Conference on Control, Automation, Robotics and Vision (Nanyang Technological University, Singapore).
- IASTED International Conference on Signal and Image Processing.

The following are journal publications:

- IEEE Transactions on Pattern Analysis and Machine Intelligence.
- IEEE Transactions on Fuzzy Systems.
- IEEE Transactions on Systems, Man and Cybernetics.
- Australian Journal of Intelligent Information Processing Systems.
- Pattern Recognition Letters.
- Computer Vision Graphics Image Processing.
- Graphical Models and Image processing.
- Electronics Letters.

2.5 Neural Network Theory

DARPA defines a **neural network** as "a system composed of many simple processing elements, operating in parallel, whose function is determined by network structure, connection strengths, and the processing performed at computing elements or nodes" [DARPA, 1988, p.60]. To this, may be added Caudill and Butler's definition [1992a] that a neural network is an information processing system which is non-algorithmic and non-digital. In current literature, there are diverse terminologies for the neural network and its components. Part of the problem lies with the close relationship to neuroscience, from which neural network draws its inspiration and paradigm. To merely adopt vocabularies common to neuroscience seems to exhibit a careless discrimination of the vast scale of qualitative difference between a neural network and a biological neural network. For this reason, in the works of Bezdek [Bezdek and Pal, 1992], Lippmann [1987], Pao [1987], Simpson [1990] and many other proponents of neural networks, we find "neurons" is labelled "computational units" or "processing elements", and "neural networks" is qualified with terms like "artificial" or "computational". In this thesis, a neural network will mean an artificial network such as a computational neural network.

Neural networks are popularly believed to have certain advantages over low level pattern recognition techniques, particularly in the following areas:

1. High processing speed.
2. Able to adapt to new information.
3. Tolerant to faults, missing information and noisy data.
4. Robust to system failure and degrades gracefully.
5. Able to generalise new patterns, where similar inputs produce similar outputs.
6. Classifies optimally.
7. Extracts model from data.

These advantages should not be treated with uncritical acceptance. Neural networks are not yet capable of modelling complex brain activities such as cognition, reasoning, memory, perception and thought. It is not even clear whether this low level of abstraction is an adequate model of the biological counterpart. A more practical problem pertaining to

of the multilayer perceptron neural network encouraged the development of other networks and established the multilayer perceptron as one of the popular models today. This neural network is known by alternative names such as the *feed forward backpropagation* or the *backpropagation*.

The architecture of a neural network typically consists of weighted connections of processing elements systematically linked together as shown in Fig. 2.15.

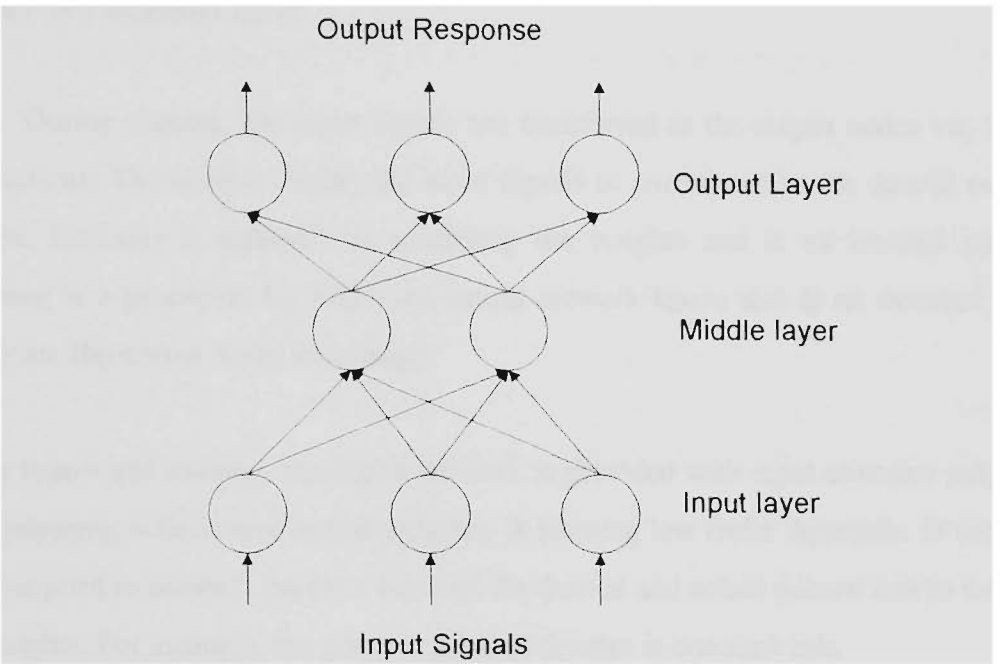


Figure 2.15 A typical architecture of a neural network. The number of connections and processing units, the physical layout of connections and the number of layers vary according to the specific neural network model. Arrow heads indicate the direction of signal flow. Circles represent the processing units. Each directed line is associated with a weight, except at the input layer.

Each processing element or unit shown in Fig. 2.15 performs three functions. In the first step, input to each unit consists of the weighted sum of the incoming signals given by

$$I_i = \sum_{j=1}^{n} w_{ij} x_j$$

where I_i is the net weighted input received by the processing unit i from a total of n units. The weight w_{ij} is the weight associated with the incoming x_j signal from the jth unit to the ith unit. In the second step, the unit receiving the I_i input converts the signal to an output given by

neural network is that the "learning" process is not transparent. Typically, it is hidden in the weight adjustments, so that the final result of training requires an act of faith to believe it is "good" enough for testing on new data. In most cases it is but, as any experimenter knows, an exception can sometimes be found. It is well known that the close interconnection of nodes and weights can cause "cross-talk" or interference between old and new patterns [Kosko, 1992]. In Chapter 8, we will demonstrate an example of learning anomalies due to cross-talk and see how a fuzzification of training data can provide dramatically improved generalisation. It is therefore prudent to recognise that a neural network has certain intrinsic limitations, the most significant of which is that there is no guarantee of error-free results, despite impressive theoretical claims of near Bayesian classification performance. Moreover, unlike a clustering algorithm, where it is possible to fix systemic errors by tuning a procedure or two, errors in the neural network cannot be so easily fixed. This is because the problem, like network learning itself, is distributed across the network. The message from this is that there is no substitute for human responsibility to validate the results of a neural network. We begin this section with a brief history and review of the structure of neural networks in Section 2.5.1. Next we examine the two main types of the fuzzy neural network designs for pattern recognition, by model embedment in Section 2.5.2 and by model mapping in Section 2.5.3.

2.5.1 Introduction

Neural networks is believed to originate from the work of McCulloch and Pits [1943]. They proposed the first model of processing units in the form of binary threshold switches and stochastic algorithms. The work by Hebb [1949] attempted to capture the concept of learning by reinforcement. In the mid-1950s and early 1960s, a class of *learning machines* emerged from Rosenblatt [1959, 1962], called *Perceptrons*. Considerable interest in the perceptrons was aroused by the mathematical proofs demonstrating convergence of **linearly separable** data to a solution in a finite number of steps. At the time however, several inadequacies soon became apparent. The basic perceptron was inadequate for most pattern recognition tasks of practical significance and lacked an effective training algorithm. A discouraging analysis by Minsky and Papert [1969] on the limitation of the perceptron stifled funds and research into neural networks until the the mid-1980s. Thereafter, a resurgence in neural network research followed the development of a new training algorithm called the "generalised delta rule for learning by backpropagation". The successful training

of the multilayer perceptron neural network encouraged the development of other networks and established the multilayer perceptron as one of the popular models today. This neural network is known by alternative names such as the *feed forward backpropagation* or the *backpropagation*.

The architecture of a neural network typically consists of weighted connections of processing elements systematically linked together as shown in Fig. 2.15.

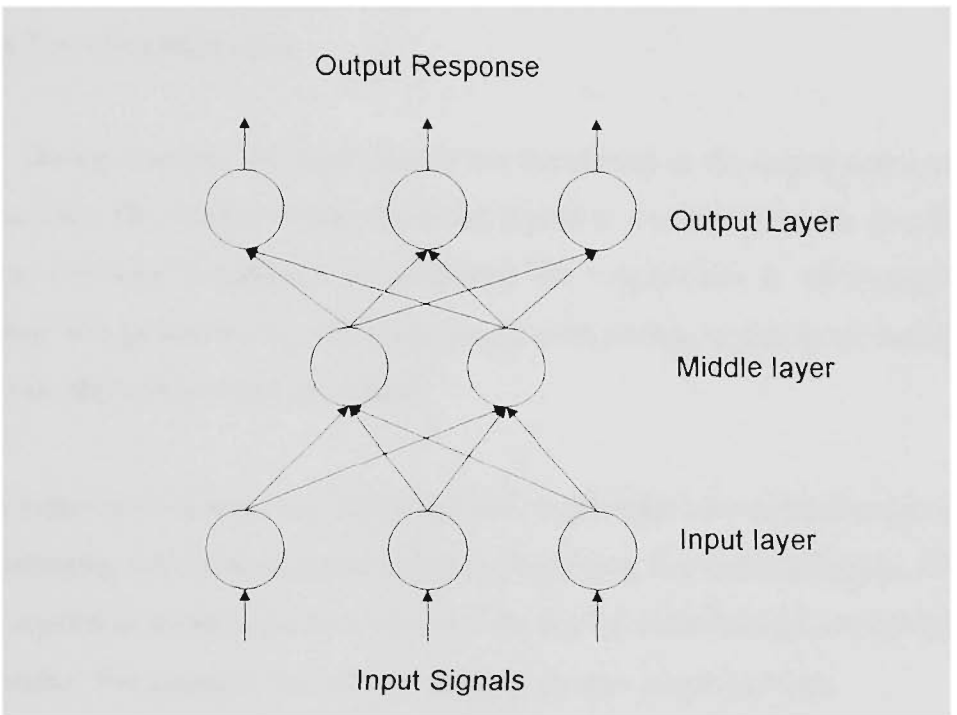


Figure 2.15 A typical architecture of a neural network. The number of connections and processing units, the physical layout of connections and the number of layers vary according to the specific neural network model. Arrow heads indicate the direction of signal flow. Circles represent the processing units. Each directed line is associated with a weight, except at the input layer.

Each processing element or unit shown in Fig. 2.15 performs three functions. In the first step, input to each unit consists of the weighted sum of the incoming signals given by

$$I_i = \sum_{j=1}^{n} w_{ij} x_j$$

where I_i is the net weighted input received by the processing unit i from a total of n units. The weight w_{ij} is the weight associated with the incoming x_j signal from the jth unit to the ith unit. In the second step, the unit receiving the I_i input converts the signal to an output given by

$$f(I) = \frac{1}{1+e^{-I}}$$

where $f(I)$ is represented by a sigmoid or S-shaped function. Other nonlinear functions can also be used, provided they are monotonically increasing and bounded with lower and upper limits. The third step performed by the processing unit is to convert the output signal to an activation level given by

$$y_i = \begin{cases} f(I) & \text{if} \quad f(I) > T \\ 0 & \text{otherwise} \end{cases}$$

where T is a threshold value.

During training, the input signals are transferred to the output nodes via weighted connections. The weights modify the input signals to correspond to the desired output response. Learning is achieved by modifying the weights and is an internal procedure. Training is a procedure by which the neural network learns and is an external process. There are three basic ways of training:

1. In supervised training, the neural network is provided with input stimulus patterns for comparing with desired output patterns. A learning law (refer Appendix D for details) is applied to compute the error between the desired and actual pattern and to modify the weights. For example, the negative gradient descent is one such rule.
2. Graded or reinforcement training provides the desired output in terms of graded levels, in contrast to the continuous levels.
3. Unsupervised training or self-organisation presents a neural network with only the input data, from which it produces an output, typically according to clustering principles.

Assuming distinct output classes, a trained network will produce a *high* output corresponding to the most likely class, while other outputs will be *low*. A taxonomy of six important neural networks identified by Lippman [1987] for classification of static patterns (in contrast to time dependent patterns) is shown in Fig. 2.16.

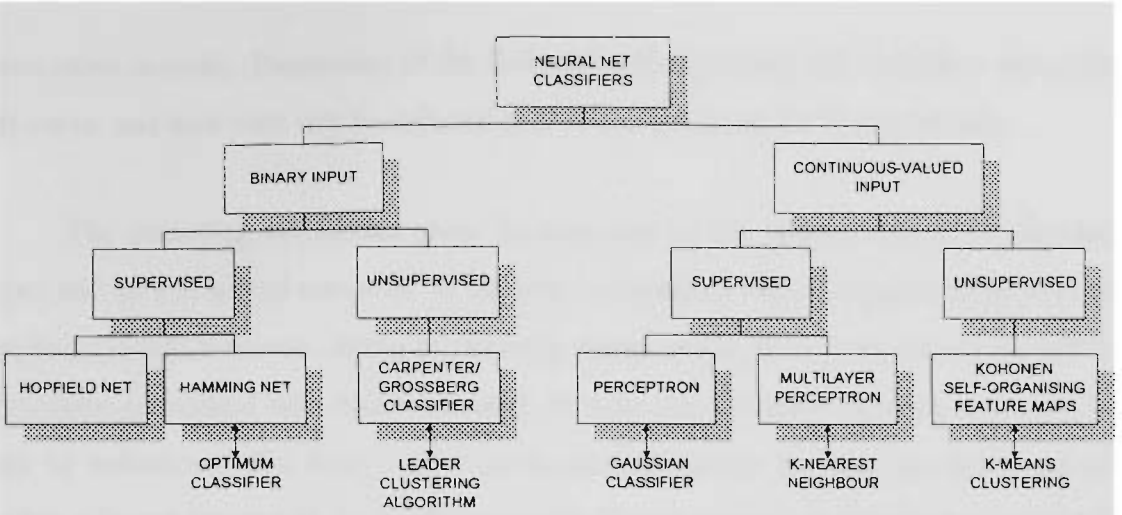


Figure 2.16 A taxonomy of six neural classifiers. Adapted from Lippman [1987].

Current research has expanded this classification of networks to include, among the more popular variety, the following: Radial Basis Function Network (RBF) [Powell, 1985], RBF with improved learning [Moody and Darken, 1989], extensions of RBF [Lee and Kil, 1991; Musavi et al., 1992], the Probabilistic Neural Network [Sprecht, 1990a and 1990b], Learning Vector Quantization (LVQ) [Kohonen, 1988, includes LVQ and LVQ2] and more advanced Adaptive Resonance Theory (ART) 2 and ART 3. The ART 1 introduced in 1986 by Carpenter and Grossberg [Pao, 1989] can only process binary inputs. The ART 2 introduced in 1987 processed gray scale input data. This was followed by the more advanced and complex ART 3 in 1989 that offers more stability in the processing of real input data patterns [Grossberg, 1988 and 1989]. Note that each of the networks has analogous statistical classifier or clustering model.

The classifiers of Fig. 2.16 are used for three different tasks: (i) as a conventional classifier (multilayer perceptron), (ii) as a content addressable or associative memory (Hopfield net) and (iii) as a vector quantiser or clustering algorithm (Kohonen self-organising). A Hopfield net [Hopfield, 1982; Hopfield and Tank, 1985a and 1985b] is normally used with binary inputs to solve optimisation problems or as a content addressable memory. As an *associative* memory network, patterns are stored by associating them with other patterns. *Autoassociative* memory stores a pattern by associating with itself. This is used to restore degraded patterns of itself. A *heteroassociative* memory stores a pattern by associating it with a different pattern. For example, an image of an ordinal number *5* might be stored by associating with the ASCII code for *5*. Other ways of classifying associative memories include the *accretive* associative memory and the *interpolative*

associative memory. Discussion of the Kohonen self-organising and multilayer perceptron networks, and their uses will be deferred to Sections 2.5.2 and 2.5.3, respectively.

The preceding discussion gives an overview of the history, nature, architecture, types and uses of neural networks. In the next two sections we will examine in more detail, the fuzzy neural networks. Being of relatively recent origin, there is no clearly identifiable paradigms associated with fuzzy networks. In most cases, a fuzzy network is obtained either by embedment of a fuzzy model for the learning law or by using the neural network within a larger framework of the fuzzy model. This is realised in practice by introducing fuzzifiers at the input end of the network and defuzzifiers at the output end or various combinations of these. These fuzzy neural networks implement fuzzy logic operations such as union (max-nets), intersection (min-nets), or the extension principle. Fuzzy networks are used to derive optimal rule sets for fuzzy controllers or to automate membership function tuning of linguistic term sets in both pattern recognition and control (see applications papers in [Bezdek and Pal, 1992]).

2.5.2 Fuzzy Model Embedment

It is well known among fuzzy pattern recognition researchers that the Kohonen Self-Organising Feature Maps (SOM) has nearly similar forms of clustering relations to FCM. Thus the integration of FCM, which is based on an optimising model, into the SOM is one way to address several limitations of SOM consisting of the following: (i) termination is based on heuristics, not on optimising a model, (ii) final weight vectors depend on input sequence and (iii) different initial conditions yield different result. The first fuzzy approach was attempted by Huntsberger and Ajjimarangsee [1989], but their scheme fell short of realising a model for the fuzzy SOM. Bezdek et al. [1992] successfully integrated the FCM model into the learning rate and updating strategies of SOM, which they named the Fuzzy Kohonen Clustering Network (FKCN). The SOM algorithm is given below [Lippmann, 1987].

Kohonen Self-Organising Feature Maps.

Step 1. Initialise weights from N input nodes to M output nodes to small random values.

Step 2. Present new input.

Step 3. Compute distance to all nodes given by

$$d_j = \sum_{j=0}^{N-1}(x_{i,t} - w_{ij,t})^2$$

where d_j is the distance from input to the jth output node, $x_{i,t}$ is the input to the ith node at iteration t and $w_{ij,t}$ is the weight from input node i to output node j.

Step 4. Select output node j^* with minimum distance, from $d_{j^*} = \min d_j$.

Step 5. Update weights to node j^* and neighbours defined by

$$w_{ij,t+1} = w_{ij,t} + \varphi_t(x_{i,t} - w_{ij,t})$$

for $j \in$ neighbourhood N_j, and $0 \le i \le N-1$. The term φ_t $(0 < \varphi_t < 1)$ is a gain term (or learning rate) that decreases in time.

Step 6. If $|w_{ij,t+1} - w_{ij,t}| < \delta$, Stop, else go to Step 2.

Note: δ is used to control the stopping point and t is an iteration index.

To apply the new fuzzy learning law, replace the weights update by the prototypes update given by

$$\mathbf{v}_{i,t} = \mathbf{v}_{i,t-1} + \varphi_{ik,t}(\mathbf{x}_{k,t} - \mathbf{v}_{i,t-1}) \tag{2.5.1}$$

for $i = 2,3,...,(N_t\text{-}1)$ where $N_t\text{-}1$ are the nodes closest to $\mathbf{x}_k$.

Next, replace the second term of (2.5.1) with a new fuzzy learning rate and prototype update to yield the iterative expression proposed by Bezdek et al. [1992]

$$\mathbf{v}_{i,t} = \mathbf{v}_{i,t-1} + \frac{\sum_{k=1}^{N} \beta_{ik,t}(\mathbf{x}_{k,t} - \mathbf{v}_{i,t-1})}{\sum_{k=1}^{N} \beta_{ik,t}} \tag{2.5.2}$$

The Fuzzy Kohonen Clustering Network algorithm (FKCN):

Step 1. Fix c.

Step 2. Initialise prototypes $\mathbf{v}_i$.

Step 3. For $t = 1,2,...,t_{max}$, compute β_{ik} and u_{ik} given by

$$\beta_{ik,t} = u_{ik,t}^{m_t}; \qquad m_t = (m_0 - 1)/t_{\max} \text{ for } m_0 > 1 \tag{2.5.3}$$

$$u_{ik} = \frac{1}{\sum_{j=1}^{c}\left[\frac{\|\mathbf{x}_k - \mathbf{v}_i\|_A}{\|\mathbf{x}_k - \mathbf{v}_j\|_A}\right]^{\frac{2}{m-1}}} \tag{2.5.4}$$

Step 4. Update prototypes from (2.5.2).

Step 5. If $\|\mathbf{v}_{i,t} - \mathbf{v}_{i,t-1}\| < \varepsilon$ (a small error), Stop, else go to Step 3.

The prototype update (2.5.2) is seen to implement the optimised prototype equation of FCM (2.4.13) as follows: Expanding the second term of (2.5.2) gives

$$\mathbf{v}_{i,t} = \mathbf{v}_{i,t-1} + \frac{\sum_{k=1}^{N} \beta_{ik,t}\mathbf{x}_{k,t}}{\sum_{k=1}^{N} \beta_{ik,t}} - \mathbf{v}_{i,t-1} \frac{\sum_{k=1}^{N} \beta_{ik,t}}{\sum_{k=1}^{N} \beta_{ik,t}}$$

which reduces to

$$\mathbf{v}_{i,t} = \frac{\sum_{k=1}^{N} \beta_{ik,t}\mathbf{x}_{k,t}}{\sum_{k=1}^{N} \beta_{ik,t}} = \frac{\sum_{k=1}^{N} u_{ik,t}^{m_t}\mathbf{x}_{k,t}}{\sum_{k=1}^{N} u_{ik,t}^{m_t}} \tag{2.5.5}$$

On comparing the results of classifying Anderson's IRIS data [Data and Hart, 1973], the FKCN showed less misclassifications compared to the SOM. The SOM always ran to its iterate limit (t_{max} = 50,000) whereas KFCN converged in 14 to 40 iterations. Consequently, there are significant benefits to be obtained from the FKCN.

The next example on embedment of fuzzy model to neural network topology is given by Sum and Chan [1994]. They adopted a similar learning strategy as Bezdek et al. [1992] but differed by introducing the FCM model for a form of the error gradient, $\frac{\partial J_m}{\partial v_i}$. The resulting prototype update equation is given by

$$\mathbf{v}_{i,t} = \mathbf{v}_{i,t} + \varphi_t \left[\sum_{j=1}^{c} \left(\frac{\|\mathbf{x} - \mathbf{v}_i\|^2}{\|\mathbf{x} - \mathbf{v}_j\|^2} \right)^{1/(m-1)} \right]^{m} (\mathbf{x} - \mathbf{v}_{i,t}) \tag{2.5.6}$$

The second term on the right hand side of (2.5.6) implements the "Delta rule" for weight update (see Appendix D), based on the FCM objective function J_m, with φ_t as a learning rate parameter of the network. Sum and Chan gave another expression that can *alternatively* be obtained by substituting $(m + 1)$ for m in (2.5.6). Consequently, the claimed "essential extension on Bezdek's definition ie. m can be set to less than 1" [Sum and Chan, 1994, p. 1851] appears illusory, and does not functionally extends FCM to the realms of $0 < m < 1$ (unlike the PFCM). Although the procedure demonstrates an interesting way to embed a fuzzy model into the learning rule, the usefulness of this approach compared to the KFCN is questionable. The output response of the neural network to a *modified* version of Ruspini's Butterfly data (incorrectly referenced as the Butterfly data in [Sum and

Chan, 1994]) exhibited very slow convergence compared to FCM, especially considering that the data contain so few points (only eleven). Using Sum and Chan's data, FCM converged in about 6 iterations (our result), but their network's convergence was still unclear at 1000 iterations (Sum and Chan's result). This means the network is about 167 times slower than FCM. The prototype values given in their paper, (0.8, 2) and (5.2, 2), at 1000 iterations, were still far (relatively speaking) from FCM's stationary points at (1.1, 2) and (4.9, 2) for $m = 2$. The slow convergence might be due to a poorly selected learning rate schedule φ_t. Furthermore, there is a possibility that the network might not converge to the same FCM stationary points. Consequently, this example shows a poor way of implementing a fuzzy model into a neural network.

2.5.3 Fuzzy Model Mapping

Neural networks used as model free estimators provide the greatest flexibility for designing a fuzzy solution compared to networks of the type reviewed in Section 2.5.2. One of the popular but reliable network used is the Feed Forward Backpropagation (FFBP). The paper by Ruck et al. [1990] indicates that any neural network, including FFBP, which incorporates a mean squared error in the learning law can be trained to approximate the Bayes optimal discriminant function for two or more classes. The FFBP network introduced by Rumelhart et al. [1986] provided a persuasive demonstration of a learning algorithm called the "Delta Learning Rule". A derivation of this learning rule is given in Appendix D. Previous work on the multi-layer perceptrons were limited to linearly separable classes. A three layer network can form arbitrary convex boundaries. The Kolmogorov theorem (described in [Lorentz, 1967]) indicates that a three layer perceptron can be used to create any continuous discriminant function in a classifier. This network structure together with a non-linear activation function and the Delta rule, provide the condition for general classification. However, practical difficulties remain in deciding how the weights, number of processing units and transfer function should be selected.

Since Lippmann's paper, improvements have been made to three areas of the FFBP: (i) training times, (ii) selection of network size and (iii) **generalisation** characteristics. These three areas are discussed in [Hush and Horn, 1993]. They showed that the problem of slow convergence is related to the complex spiral staircase-like stepped surfaces encountered in the gradient descent of a non-linear unit. Current research interest in the network's

weights selection and control is linked to strong evidence which suggest improvements to all of the three areas above. Reducing the number of weighted connections, up to a point, appear to improve training times and generalisation. Generalisation is a measure of how well the network performs on actual problems after training is completed. It is usually tested by evaluating the network's performance on new data, apart from the training set.

We will only examine two methods for weight reduction, because of interesting procedural similarity with the optimisation of the fuzzy objective function. The first method of weight reduction is called *weight decay* [Hanson and Pratt, 1989; Hinton, 1987]. This mode of weight reduction is viewed as a way of reducing the effective number of weights by encouraging the network to seek solutions that use as many zero or near zero weights as possible. The criterion function has the form

$$J(w)=\sum_{p=1}^{P}J_p(w)+\frac{\lambda}{2}\sum_i w_i^2 \tag{2.5.7}$$

where the first term is the squared error criterion and the second term is included to penalise the network for using non-zero weights. The λ factor is a small constant to control the influences of the second term relative to the first term. How does this improve generalisation? The answer is that not all weights are made smaller. Some will remain relatively large, while others will be forced under the learning rule, to be small in order to minimise the criterion function $J(w)$. The net effect is a reduction of those weights which have little influence on the solution and so improve generalisation by discouraging overfitting of data. An alternative technique is called *weight elimination* [Weigend et al., 1990 and 1991]. The criterion function is of the form

$$J(w)=\sum_{p=1}^{P}J_p(w)+\lambda\sum_i \frac{w_i^2/w_0^2}{1+(w_i^2/w_0^2)} \tag{2.5.8}$$

where w_0 is a fixed weight normalisation factor. When $w_i > w_0$, the sum approaches unity and this criterion counts the number of weights. When $w_i < w_0$, the sum is proportional to w_i^2 and the criterion works like a weight decay criterion. Thus it is seen that the choice of w_0 can influence this criterion function to encourage the network to seek solutions with a few large weights (w_0 small) or many small weights (w_0 large). Therefore (2.5.8) is a generalisation of the weight reduction scheme of (2.5.7).

Other methods for generalisation will be explored and discussed in Chapter 8. One practical way of using a neural network is to map the model of the clustering algorithm

implicitly, by training the network with labelled data. Assuming there are no problems in mapping the model of the algorithm, then it should be possible to extract cluster prototypes with a neural network. Given the prototypes, it should be possible to find the points associated with each prototype, once the algorithm has been defined. The advantage in this scheme is the significant speed improvement from a fully trained network compared to a clustering algorithm, especially where a large amount of data are involved. This might be the only viable means for real-time applications and could provide a more cost effective solution compared to a hardware implementation. Taking this process a step further, it might also be possible for the network to model an expert's knowledge of an ideal clustering algorithm *without* the need for an actual algorithm to be implemented. This would be a highly desirable goal, but seems beyond the reach of present achievements. For the moment, many problems are yet to be resolved because the simplest and most direct method of modeling the algorithm has not been successful. However, indirect methods involving fuzzy memberships, fuzzifier and defuzzifier stages appear promising as a way of inducing meaningful cluster prototypes from the network's response. We discuss examples of this approach in Chapter 8.

Remarks

A good general introduction to neural networks is found in [Lippmann, 1987; Hush and Horn, 1993]. Numerous textbooks giving practical insights into specific network topologies and applications are to be found in [Freeman and Skapura, 1992; Kosko, 1992; Zurada, 1992; Hertz et al., 1991; Simpson, 1990; Pao, 1989]. Some of these include software programmes to experiment on PC compatible computers [Rao and Rao, 1995; Welstead, 1994; Masters, 1993]. An interesting and surprisingly effective way to experiment with neural network is network modeling with Mathematica in [Freeman, 1994]. Alternatively, but generally more complicated, is the use of professional neural software packages such as the HNC ExploreNet 3000, release 2.12, or the NeuralWorks professional II/Plus and NeuralWorks Explorer from NeuralWare Inc. Both of these last two products include informative descriptions of the main network types. There is generally quite a steep learning curve to overcome initially, but once over this curve, the ability to do more with neural networks are well rewarded. A more affordable approach with good tutorials on neural networks are the two volumes by Caudill and Butler [1992], which contain neural softwares for the Macintosh and IBM PC compatibles. Comprehensive information on fuzzy

neural networks is found in [Bezdek, 1995a and 1995b; Bezdek and Pal, 1992]. Conference papers and journal publications relating to fuzzy clustering and pattern recognition topics are listed in "Remarks" of Section 2.4.

Chapter 3

An Enhanced Progressive Fuzzy Clustering Approach

The theory development for fuzzy clustering approaches to pattern recognition has been presented in Chapter 2. In particular, we have reviewed the analytic forms of the different fuzzy objective functions with and without constraints, and showed how optimised solutions for the cluster parameters can be obtained from an iteratively minimised objective function exemplified by the FCM algorithm. Lastly, we examined procedures to formulate clustering solutions that can be generalised for a range of ellipses with the fuzzy covariance matrix, and for hollow rings and ellipses using the method of parametrized prototype.

In this chapter, we shall apply the basic attributes of fuzzy analytic models in a new way that enhances its scope and potential. We call this the enhanced progressive fuzzy clustering approach or CPCM. In the chapter we establish the rationale and justification to link the analytic fuzzy solutions to the CPCM framework. Although the CPCM approach is to some extent dependent on the fuzzy analytic theory and models to produce useful results, it does not require the level of theory needed in those algorithms to harness the power of fuzzy clustering. We demonstrate how a judicious selection of a few basic equations is usually effective for this purpose. The CPCM framework takes care of the intricate details of supervising cluster development.

3.1 Introduction

The CPCM approach to fuzzy clustering introduces five new features to fuzzy clustering, consisting of the following:

1. Progressive clustering on a single cluster prototype assumption.
2. Cluster parameter prescription by structural specifications such as minimum number of points in a cluster, N_{min} and the alphacut, α_t.
3. More efficient management of data and the clustering process.
4. More flexible framework to generate cluster solutions.
5. Improved useability and utility.

Clustering on a single prototype assumption means progressive clustering for a single cluster. There are advantages with progressive clustering. Firstly, the principle of clustering the entire data set is analytically more complex; consequently, more sensitive to other factors influencing the outcomes of the cluster solutions, such as noise and outliers, and the structure of neighbouring clusters. This is evident in the sensitivity of the conventional fuzzy clustering results to initial conditions for U_0 and V_0 (see Krishnapuram and Keller, 1993a; Man and Gath, 1994; Gath and Hoory, 1995). Conceivably, a single cluster is a simpler representation of data for fuzzy clustering. Moreover, the algorithm is isolated from extraneous influences of neighbouring data so that it is easier to represent the typical characteristics of a single cluster. To realise this goal in CPCM, it is not possible to apply the fuzzy algorithms without some modifications. Furthermore, the modified algorithm needs to be constrained to look for only a single cluster. We will show how these issues are resolved in Section 3.2.

The cluster parameter prescription by structural specifications serves two objectives. Firstly, it eliminates the need for cluster size verification because the specifications will influence the development of clusters and determine the number of clusters found. This means greater computation efficiency compared to those algorithms that require cluster validity checks. Furthermore, the automatic detection of cluster size feature is a simpler, more appealing and intuitive approach to clustering. Secondly, the cluster structural parameters such as the alphacut α_t and the minimum cluster points N_{min} have significant and unique implications for cluster development. Unlike the conventional fuzzy clustering schemes, this approach provides a high degree of cluster generalisation. In other words, the cluster parameters help to identify structurally similar clusters in data. This feature is not available to algorithms such as the FCM or the KNN, because the initial c prescription pre-empts the actual cluster numbers present in the data. The CPCM parameters not only

provide a means of control over cluster development, but also encourage an engineering design approach to solve clustering problems. For example, α_t determines the fuzziness of the cluster boundary and N_{min} provides a design criterion which determines whether small or big clusters will be developed.

The efficient management of data and the clustering process is important to any clustering algorithm because this may have a significant impact on the processing times. A progressive clustering scheme has greater computation efficiency compared to a non-sequential or global clustering scheme. This occurs primarily because each cluster that is found can be immediately eliminated from subsequent cluster consideration. Thus as clustering proceeds, the data present is progressively reduced, resulting in greater computation efficiency. The more data that is removed earlier, the greater will be the computation efficiency. However, the sequential nature of clustering may produce fragmented clusters if the clusters are not compact and well separated. Therefore, remedial measures to ensure good outcomes need to be considered. These issues are examined in Section 3.2.3.

CPCM provides a flexible framework to generate cluster solutions. CPCM supervises the management of data relating to cluster validity and noise, freeing the clustering algorithm from these tasks. CPCM also handles the clustering context so that the clustering algorithm "sees" only a single cluster. This avoids compatibility issues and allows analytic algorithms to be adapted into the CPCM framework.

Improved useability and utility of the CPCM based algorithms are obtained from the improvements offered by each of the above mentioned features (items one to four). The ability to integrate specific clustering algorithms within a common CPCM framework enhances clustering performance and simplifies the task of algorithm development. An example of this is given in Section 3.4, involving the detection of clusters with different shapes and types.

These five items are discussed more comprehensively in the following sections of this chapter. Section 3.2 introduces the principles of clustering within the CPCM framework, followed by a cluster validity reviewed in Section 3.3. Some experimental algorithms are explored in Section 3.4 which shows how to apply the fuzzy clustering equations within

the CPCM framework. The chapter concludes with performance comparisons of FCM, KNN and the CPCM based algorithms in Section 3.5.

3.2 Clustering Principles

Clustering with CPCM generally involves the iterative update of two basic cluster parameters, consisting of the cluster membership and the prototype discussed below:

3.2.1 Membership Update

The membership equation for a given cluster is similar to the form of (2.4.50) given by

$$u_k = \frac{1}{\exp(d_k^q / \eta_j)} \tag{3.1}$$

where

$$d_k = \|\mathbf{x}_k - \mathbf{v}_j\| \tag{3.2}$$

is the distance of a data point $\mathbf{x}_k$ from its jth cluster prototype $\mathbf{v}_j$. The data $\mathbf{x}_k$ denotes the kth item of the data set, $X = \{\mathbf{x}_1, \mathbf{x}_2, .. \mathbf{x}_k, ..., \mathbf{x}_N\}$ in real feature space $\Re^d$. In the experimental algorithms to be presented shortly, q is assumed unity to simplify calculations. The lower its value, the fuzzier is the boundary of the cluster. The η_j term in (3.1) is included to give a meaningful value to the membership. Since it is like a scale factor, it can also be defined as a root mean square of the cluster radius

$$\eta_j = s\sqrt{\frac{1}{N_j}\sum_{k=1}^{N_j} |\mathbf{x}_k - \mathbf{v}_j|^2} \tag{3.3}$$

where N_j is the number of feature vectors that satisfy α_t in the jth cluster for $0 < j < N$, and s is a scale factor. Equation (3.3) scales the memberships of (3.1). In some cases a fixed value of η_j may also be used (see examples in Chapter 6).

3.2.2 Prototype Update

The prototype update equations are identical to FCM, except for the single cluster assumption. It is expressed by

$$\mathbf{v}_j = \frac{\sum_{k=1}^{N_j} u_k^m \mathbf{x}_k}{\sum_{k=1}^{N_j} u_k^m} \tag{3.4}$$

In most cases, a satisfactory choice for m is 2. A practical version of (3.4) that reflects the alphacut α_t is given by

$$\mathbf{v}_j = \frac{\sum_{k=1}^{N_j} u_k^2 \mathbf{x}_k F_k}{\sum_{k=1}^{N_j} u_k^2 F_k} \tag{3.5}$$

$$F_k = \begin{cases} 1 & \text{if} \quad u_k > \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

A general iterative procedure for clustering is given by the following algorithm:

A general CPCM clustering algorithm.

Step 1. Fix α_t, m, q, s, N_{min}.

Step 2. Compute initial prototype $\mathbf{v}_0$ from (3.5) for $u_k = 1$.

Step 3. Compute η_j from (3.3) and d_k from (3.2).

Step 4. Calculate memberships from (3.1).

Step 5. Calculate prototypes from (3.5).

Step 6. Repeat Steps 3 to 6 until $\|\mathbf{v}_{j,t} - \mathbf{v}_{j,t-1}\| < \varepsilon$.

Note: ε is a small value to control the stopping point and t is an iteration index.

If there are more than two cluster parameters to be solved, the basic iterative algorithm remains the same, except to allow for the extra parameters in the metric. For example, to detect circular rings of fixed diameter, a radius parameter is included in the general metric of (3.2). See Chapter 6 for variable ring detection.

3.2.3 The CPCM Framework

The CPCM algorithm provides the framework to supervise the management of data with respect to initial prototype estimation and to provide the impetus to sustain a progressive search for clusters in the data set. The latter includes tests for cluster validity.

3.2.3.1 Estimating Initial Prototype

Since CPCM assumes a single cluster for prototype development, it is necessary to obtain an initial estimate of the prototype for use by the clustering algorithm. There are several ways of doing this but we will only examine two of these.

(A) Sequential Fuzzy Means (SFM)

Derived from the basic FCM, this algorithm has not only a surprising simplicity but also an unusual prototype solution. Although it involves only *one* cluster parameter (the metric), it is nevertheless capable of producing *two* solutions for the cluster prototype. Since this appears to be a new discovery (to the best of our knowledge), we will call it the SFM. Perhaps one reason why it is not found earlier may be attributed to the implausible notion that derives sequential clustering from FCM. To recapitulate, we recall the FCM membership

$$u_{ik} = \frac{1}{\sum_{j=1}^{c}\left(\frac{d_{ik}^2}{d_{jk}^2}\right)^{\frac{1}{(m-1)}}} \tag{3.6}$$

where the numerator term is unity because of the constraint assumption $\sum_{i=1}^{c} u_{ik} = 1$ (see Appendix A for details of the proof). The form of (3.6) is obviously not valid for a single cluster since for $c = 1$, $u_{ik} = 1$ for *all* points of the data set. However, eliminating the d_{ik} term from (3.6) with the constraint (assuming $c > 1$)

$$\sum_{i=1}^{c} u_{ik} = d_{ik}^{2/(m-1)} \tag{3.7}$$

yields a new membership

$$u_{ik} = \frac{1}{\sum_{j=1}^{c}\left(\frac{1}{d_{jk}^2}\right)^{\frac{1}{(m-1)}}} \tag{3.8}$$

from which it is apparent that (3.8) will admit a single cluster solution given by

$$u_k = \frac{1}{\left(\frac{1}{d_k^2}\right)^{\frac{1}{(m-1)}}} \tag{3.9a}$$

or

$$u_k = d_k^{2/(m-1)} \tag{3.9b}$$

where $d_k = \|\mathbf{x}_k - \mathbf{v}\|_A$, and u_k being roughly proportional to the metric now represents the dissimilarity index (without upper bound) for the constraint of (3.7) ie. prototypes are located at min u_k. Under the single cluster assumption, (3.9b) is seen to be equivalent to (3.7). The prototype equation for this case is identical to (3.4). Note that (3.9b) may also be generalised by PFCM. The SFM algorithm is similar to FCM except for the membership (3.9b) and using $c = 1$.

To illustrate the interesting properties associated with (3.9b) we present the SFM cluster prototype convergence characteristics shown in Fig. 3.1, for the data of Fig. 2.11, and the membership results in Tables 3.1 and 3.2 (one for each prototype).

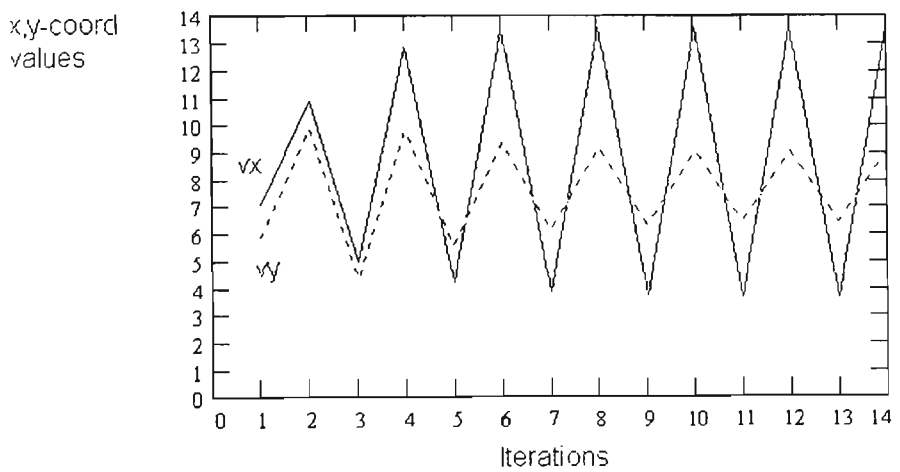


Figure 3.1 SFM prototype convergence for data set of Fig. 2.11 at $m = 1.6$. Prototype positions in the x and y axes are represented by vx and vy respectively, defined at the upper and lower envelopes of each curve. The correct prototype converged to (13.58, 9.02).

k	u_k	k	u_k	k	u_k	k	u_k
1	4633	7	4.62	13	620	19	1497
2	3558	8	1.56	14	1933	20	2991
3	3516	9	1.69	15	1471	21	2434
4	3561	10	1.25	16	1115	22	1993
5	2602	11	1.37	17	2401		
6	1138	12	3.20	18	1894		

Table 3.1 SFM memberships from data set of Fig. 2.11 for prototype (13.58, 9.02) at $m = 1.6$. Minimum membership is shown shaded.

k	u_k	k	u_k	k	u_k	k	u_k
1	76.47	7	1384	13	60.27	19	444.15
2	95.00	8	2175	14	117	20	377.13
3	40.67	9	1818	15	188.19	21	485.38
4	15.13	10	2932	16	304.37	22	648.08
5	25.51	11	2527	17	217.77		
6	1151	12	3593	18	306.36		

Table 3.2 SFM memberships from data set of Fig. 2.11 for prototype (3.64, 6.41) at $m = 1.6$. Minimum membership is shown shaded.

Fig. 3.1 shows SFM converging in about 6 iterations to two stationary points for the data of Fig. 2.11. Clearly the prototype at (13.58, 9.02) is the correct prototype, whilst the other prototype may be ignored since only one cluster is under consideration. This may be easily verified by checking the memberships in proximity to the prototype. Also, prototype (13.58, 9.02) has least membership at $u_k = 1.25$, whereas prototype (3.63, 6.43) has higher minimum membership at $u_k = 15.13$. The memberships of Tables 3.1 and 3.2 confirm that the prototypes are located at points of minimum membership given by (3.9b).

It is interesting to observe that the SFM convergence *always* produces two prototypes in alternating order within the same sequence. This feature could be usefully exploited for the two clusters of the Butterfly data set. The results are given in Fig. 3.2 and Table 3.3. Notice that *both* the two prototypes at (1.818, 3) and (6.182, 3) are found correctly, for $m = 2.72$ (other m values shift the prototype position) and is almost identical to FCM's prototypes at (1.855,3) and (6.145,3), for $m = 2$. As noted previously, the two prototypes generated by SFM are both valid only for a two-cluster data set.

Overall, SFM is expected to work better on well separated compact clusters than on close sparse clusters. Although more research is needed to establish the conditions for good clusters, SFM is presented here as an alternative and useful means of pursuing sequential fuzzy clustering with improved computation efficiencies compared to global clustering schemes. SFM could also be useful in another role. Earlier we mentioned about remedial measures to overcome poor cluster result. One method is to select dense clusters with SFM during initial clustering (to minimise cluster fragmentation).

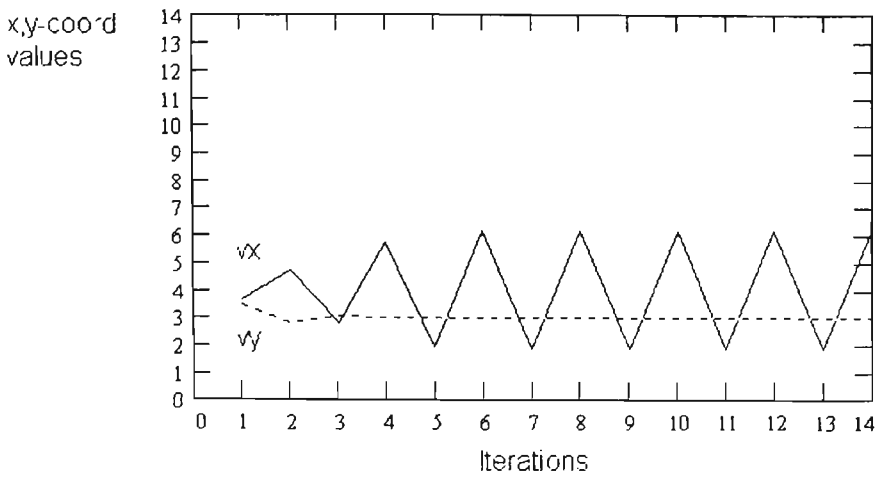


Figure 3.2 SFM prototype convergence for the data set of Fig. 2.5 at $m = 2.72$. Prototype positions in the x and y axis are represented by vx and vy respectively, defined at the upper and lower envelopes of each curve. The two prototypes converged to (1.818, 3) and (6.182, 3).

k	u_k	k	u_k	k	u_k	k	u_k	k	u_k
1	2.449	4	1.019	7	1.215	10	5.453	13	7.344
2	0.791	5	0.138	8	2.478	11	5.280	14	6.774
3	2.449	6	1.019	9	3.842	12	5.453	15	7.344

Table 3.3 SFM memberships from the data set of Fig. 2.5 for prototype (1.818, 3) at $m = 2.72$. Memberships for prototype (6.182, 3) is symmetrically opposite.

(B) Data Centroid

In this scheme, the data centroid is computed from (3.4), for $u_k = 1$. Next, the complete data set is searched for the nearest neighbour to the prototype. This position is used as the initial prototype. In the event of a tie, the nearest neighbour point is resolved arbitrarily (eg. taking the first occurrence). The nearest neighbour locates the prototype at a real data point in X. CPCM applies the test $N_\alpha > N_{min}$ to ensure that only a valid cluster is found.

3.2.3.2 Sustaining the Progressive Clustering Cycle

Progressive clustering is realised by a cycle of activities involving: (i) initial prototype centring around the data centroid, (ii) developing a cluster, (iii) verifying a cluster, (iv) removal of cluster or non-cluster points, and (v) update of the data list. CPCM initiates each clustering cycle until the data set is depleted below the level of the minimum cluster points specification, N_{min}. A valid cluster satisfies the specifications for α_t and N_{min}. Cluster

membership is defined by equation (3.1). An option to consider at this point is to recluster the data using the most recent cluster prototype positions. This procedure minimises cluster fragmentation but compromises computation efficiency.

A basic CPCM framework.

Fix α_t, m, q, s, N_{min}.

Repeat

Assume initial $\mathbf{v}_0$ from nearest neighbour of data centroid.

Repeat

Calculate u_k from (3.1).

Calculate $\mathbf{v}_t$ from (3.5).

Until $\|\mathbf{v}_t - \mathbf{v}_{t-1}\| < \varepsilon$.

If $(0 \le N_\alpha < N_{min})$ **Then** Remove N_α and update N_c.

If $(N_\alpha \ge N_{min})$ **Then** Save and remove cluster points of prototype $\mathbf{v}$ and update N_c.

Until $(N_c < N_{min})$.

Note that ε is a small value to control the stopping point and t is an iteration index.

The symbols N_α, N_{min} and N_c represent respectively, the number of cluster points obtained from applying alphacut α_t to u_k, the minimum number of points defining a valid cluster, and the current data count. In the experimental algorithms, alphacut α_t has values greater than 0.9 and the fuzzy factor q is assumed unity. Note that the inner "repeat-until" loop can include *any* compatible clustering algorithm or support more than one algorithm. Examples of this are given in Section 3.4.

Effective use of CPCM requires a judicious selection of cluster parameters α_t and q. They are normally determined empirically. The alphacut α_t determines the size of the cluster region and the fuzziness of the cluster boundary. The impact of α_t on cluster development will be explained in Section 3.4 and Chapter 5.

3.3 A Cluster Validity Measure

For the purposes of evaluating the experimental algorithms and comparing their performance against standard algorithms like the FCM and the KNN, we define a point cluster va-

lidity measure as follows (see Section 2.3.4 for other ways of defining cluster validity). For each object k of cluster A, we calculate

$$a(k) = d(k, v_k) \tag{3.10}$$

where $d(k,v_k)$ is the distance of k from its cluster prototype v_k, assuming cluster A is not a singleton. Next we calculate the distance of object k from the *nearest* cluster prototype in cluster C, where cluster $C \neq A$, given by

$$b(k) = \min_{k \neq j} d(k, v_j) \quad \text{for} \quad j = 1, \ldots, c \tag{3.11}$$

where c is the number of clusters in the data set. The cluster structure coefficient or index $s(k)$ is calculated by combining (3.10) and (3.11) as

$$s(k) = 1 - \frac{a(k)}{b(k)} \quad \text{if} \quad a(k) < b(k) \tag{3.12a}$$

$$s(k) = 0 \quad \text{if} \quad a(k) = b(k) \tag{3.12b}$$

$$s(k) = \frac{b(k)}{a(k)} - 1 \quad \text{if} \quad a(k) > b(k) \tag{3.12c}$$

The $s(k)$ index is a measure of the structure of the clusters. If cluster A contains only one object, it is unclear how $a(k)$ is defined so we assume arbitrarily $s(k) = 0$ in (3.12b). Other values of $s(k)$ are possible. Note that (3.12a) indicates that if k is closer to its own prototype than to its nearest neighbour prototype, then $s(k)$ is positive (max is +1), and makes a positive contribution to the similarity index. If k is closer to its neighbour prototype than its own prototype, as in (3.12c), then $s(k)$ is negative (min is −1), indicating a penalty. If k is equi-distance from either cluster prototypes, then $s(k) = 0$ and makes no contribution to the similarity index because the point lies on the decision boundary. Thus the range of values for $s(k)$ are: $-1 \leq s(k) \leq 1$. When $s(k)$ is close to 1 we say k is well classified ie. the proximity of k to its own prototype is unambiguous. We may also define cluster measures for each jth cluster $\bar{s}_1(j)$, and for the entire data set $\bar{s}_2$ as follows.

$$\bar{s}_1(j) = \frac{1}{N_j} \sum_{k=1}^{N_j} s(k) \tag{3.13}$$

$$\bar{s}_2 = \frac{1}{N} \sum_{k=1}^{N} s(k) \tag{3.14}$$

where N_j is the number of points in the jth cluster, and N is the total number of points in the data set.

3.4 Experimental Algorithms

The experimental algorithms presented in this section are intended to demonstrate the feasibility of adapting the fuzzy analytic clustering algorithms to the CPCM framework. This discussion should not be construed to imply that the experimental algorithms introduced in this section are of little practical significance or unsuitable for real world applications. We feel there are potentially many useful ideas emerging from the experimental algorithms than could be adequately considered in this chapter. Obviously, algorithms designed for practical applications require more thorough and extensive treatments of a practical nature. We shall examine this issue more fully in chapters 4 to 7.

3.4.1 Round Cluster Structure

Round regions are possibly one of the most common cluster structure in existence and also one of the most useful structure from an application perspective. Round regions are detected with the membership and prototype equations given by

$$u_k = \frac{1}{\exp(d_k / \eta)} \tag{3.15}$$

$$\mathbf{v}_j = \frac{\sum_{k=1}^{N_j} \mathbf{x}_k}{N_j} \tag{3.16}$$

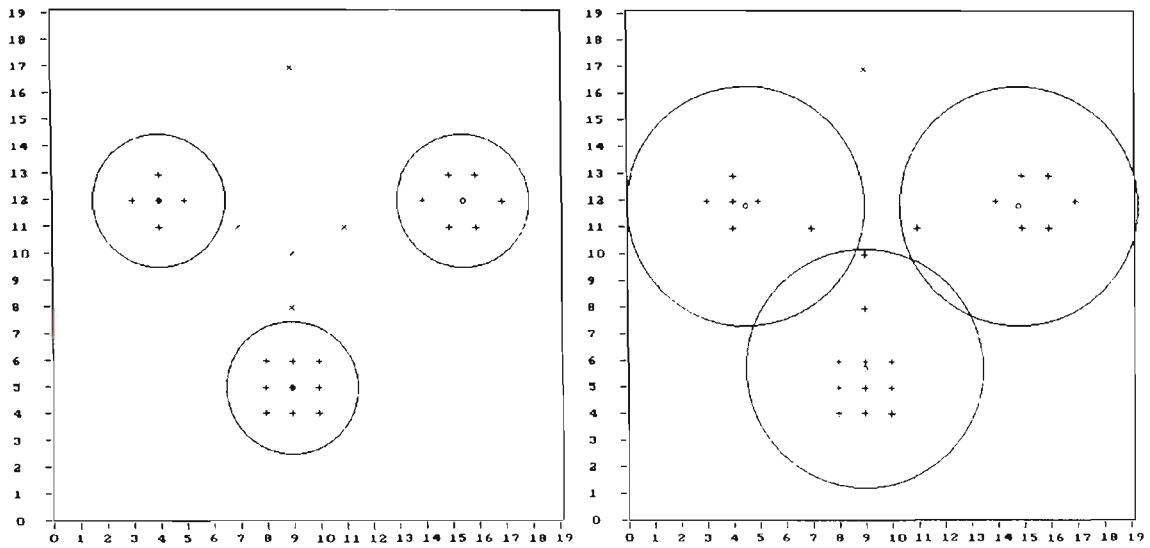
where N_j is the number of points in the jth cluster having cluster prototype, $\mathbf{v}_j$. The cluster prototype $\mathbf{v}$ and the data point $\mathbf{x}$ are feature vectors in $\Re^2$. Applying the alphacut α_t to Eq. (3.15), results in a fixed cluster radius

$$r_k = \frac{-\eta \ln \alpha_t}{2} \tag{3.17}$$

where $d_k = 2r_k$ and defines the boundary of a circular region for a fixed η and α_t

These equations are sufficient to produce centroid seeking prototypes via a clustering mechanism unique to CPCM. Iterations of (3.15) and (3.16) for a region defined by (3.17) generate a *dynamic* (moving) circle that iteratively centres itself around the centroid of points contained in the region defined by (3.17). The prototype is defined by points whose memberships exceed the specified α_t level. This interesting clustering characteristic discloses an appealing geometric interpretation of the fuzzy cluster and provides a simple

mechanism for clustering that converges to a local centroid. For this reason we named this basic clustering process, cluster prototype centring by membership or CPCM.



(a) (b)

Figure 3.3 Three clusters detected from a data set of 25 points (modified from Kaufman and Rousseeuw's data set of Fig. 2.11). (a) is obtained with $r_k = 2.5$ and (b) with $r_k = 4.5$. In both cases, the minimum cluster points, $N_{min} = 5$. Note: "+" = cluster point and "×" = non-cluster point.

Figure 3.3 shows the effects of clustering with $r_k = 2.5$ units and $r_k = 4.5$ units (delineated by the circular boundaries). The small circle denotes the cluster centre. Notice that the prototypes of Fig. 3.3(a) are located at the local centroids (4, 12), (9, 5) and (15.5, 12) whereas the prototypes of Fig. 3.3(b) is slightly off-centroid at (4.5, 11.83), (9, 5.73) and (14.86, 11.86), because of the influence of outliers near the data centroid. In the context of the CPCM algorithm, point (9, 17) which is unclustered in Figs. 3.3(a, b), may be regarded as a virtual noise point. However, making r_k bigger will eventually cluster this point. In contrast to conventional fuzzy clustering algorithms, a significant advantage of CPCM is the automatic detection of the number of clusters demonstrated in Fig. 3.3. Moreover, CPCM is also designed to detect clusters of a specified radius r_k, an advantage in certain applications where the cluster diameter or area can be approximated from the problem domain.

We know that the CPCM algorithm detects clusters with a fixed size. Can it be made to detect *variable* cluster sizes like the FCM? The answer is yes, by allowing for variable r_k or η. Figures 3.4 and 3.5 are examples of a CPCM algorithm using a variable η. The cluster statistics are summarised in Table 3.4.

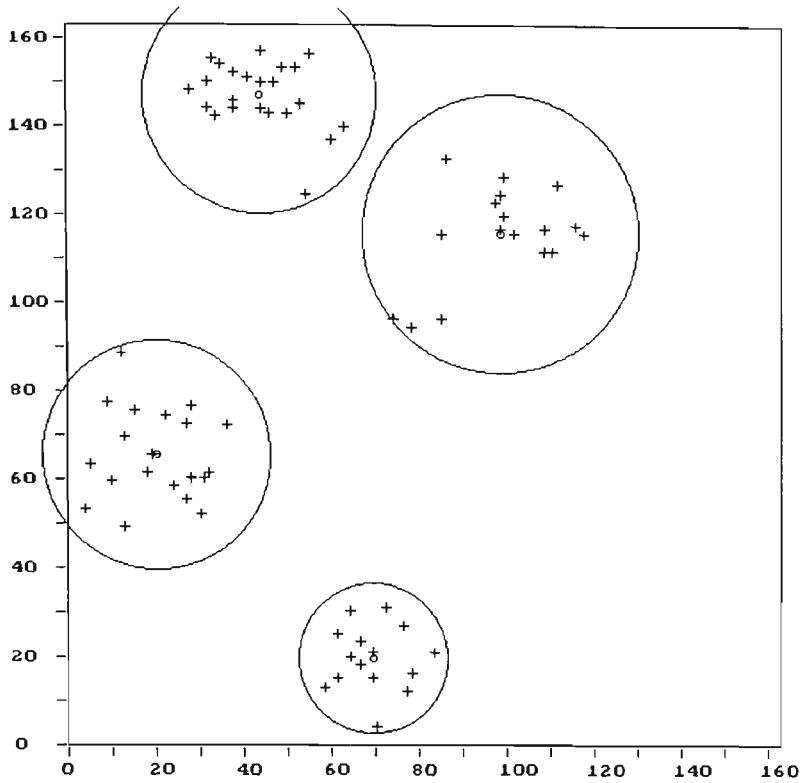


Figure 3.4 Four clusters detected from the Ruspini's data set, using variable η_j. The cohesion factor, $f_c = 9$. Minimum cluster radius, $r_{min} = 17$. Positions of the data points are given at the end of Appendix F.

Cluster	Centre (pixel units)	Radius (pixel units)	Cluster index, $\bar{s}_1$	N_α
1	68.93, 19.40	17.00	0.8664	15
2	20.15, 64.95	25.58	0.8070	20
3	43.91, 146.04	26.46	0.8222	23
4	98.18, 114.88	31.10	0.7727	17

Table 3.4 Summary of cluster statistics from the data set of Fig. 3.4, for a cohesion factor $f_c = 9$ and $r_{min} = 17$. Data set cluster index $\bar{s}_2 = 0.8158$. N_α is the number of points in the cluster.

A minimum cluster radius r_{min} is specified to prevent a cluster shrinking (via cluster convergence) to zero radius. A minimum η corresponding to r_{min} is given by

$$\eta_{\min} = -\frac{2r_{\min}}{\ln \alpha_t} \tag{3.18}$$

At the start of the cluster repeat-until loop, the variable η_j of the jth cluster is calculated as

$$\eta_j = \begin{cases} \dfrac{f_c \sum_{k=1}^{N_j} \|\mathbf{x}_k - \mathbf{v}_j\|}{N_j} & \text{if} \quad \eta_j > \eta_{\min} \\ \eta_{\min} & \text{otherwise} \end{cases} \tag{3.19}$$

where f_c is a cohesion factor that relates to the size and number of clusters formed. A high f_c results in bigger cluster radii but smaller number of clusters, and vice versa. Within the cluster algorithm, the cluster memberships and prototypes are computed from (3.15) and (3.16) repeated below:

$$u_k = \frac{1}{\exp(d_k / \eta_j)} \tag{3.20}$$

$$\mathbf{v}_j = \frac{\sum_{k=1}^{N_j} \mathbf{x}_k F_k}{\sum_{k=1}^{N_j} F_k} \tag{3.21}$$

$$F_k = \begin{cases} 1 & \text{if} \quad u_k > \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

where the number of points in the jth cluster is calculated from

$$N_j = \sum_{k=1}^{N_c} F_k \tag{3.22}$$

The CPCM algorithm for variable η_j and variable cluster sizes.

Fix α_t, q, r_{min} and N_{min}.

Repeat

Assume data centroid for initial $\mathbf{v}_0$.

Calculate η_j from (3.19).

Repeat

Calculate u_k from (3.20).

Calculate $\mathbf{v}$ from (3.21).

Calculate η_j from (3.19).

Until $|\eta_{j,t} - \eta_{j,t-1}| < \delta$.

If $(0 \le N_\alpha < N_{min})$ **Then** Remove N_α and update N_c.

If $(N_\alpha \ge N_{min})$ **Then** Save and remove cluster points of prototype $\mathbf{v}$ and update N_c.

Until $(N_c < N_{min})$.

Note: δ is a small value to control the stopping point and t is an iteration index.

Figure 3.4 shows four clusters obtained for a cohesion factor $f_c = 9$. Like the fixed cluster radius algorithm, the variable η_j algorithm also finds the cluster numbers automatically. Unlike the previous algorithm, it finds the cluster radius automatically. Figure 3.5 shows

three clusters from the same data set for $f_c = 17$. The corresponding cluster statistics are given in Table 3.5.

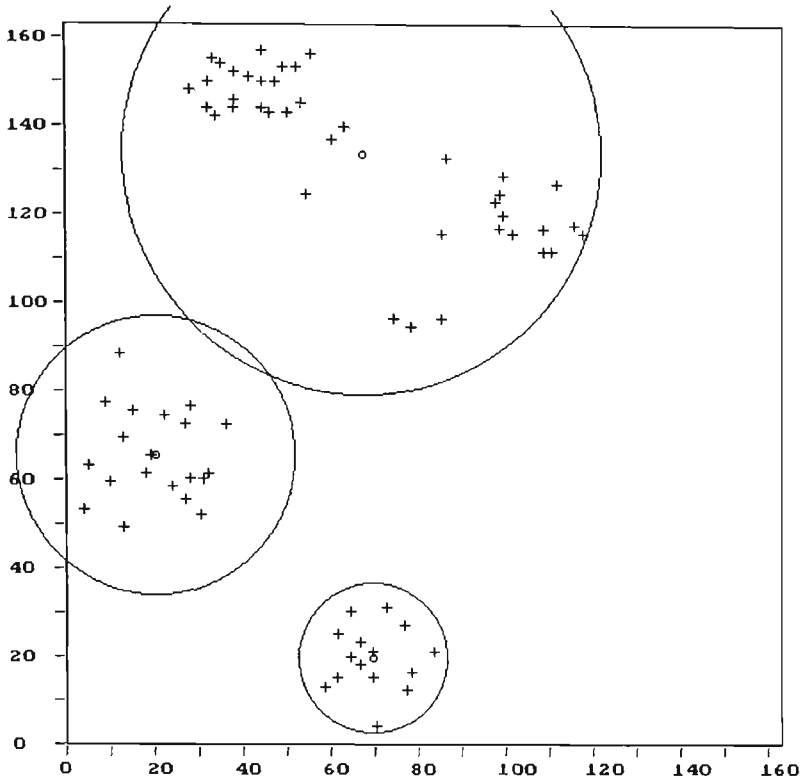


Figure 3.5 Three clusters detected from the data set of Fig. 3.4, using variable η_j. The cohesion factor, $f_c = 17$. Minimum cluster radius, $r_{min} = 17$.

Cluster	Centre (pixel units)	Radius (pixel units)	Cluster index, $\bar{s}_1$	N_α
1	68.93, 19.40	17.00	0.8664	15
2	20.15, 64.95	31.32	0.8086	20
3	66.97, 132.8	54.12	0.6338	40

Table 3.5 Summary of cluster statistics from the data set of Fig. 3.4, for a cohesion factor $f_c = 17$ and $r_{min} = 17$. Data set cluster index $\bar{s}_2 = 0.7269$. N_α is the number of points in the cluster.

The definition of the cluster index $\bar{s}_1$ is biased towards a smaller number of compact clusters while the data set cluster index $\bar{s}_2$ favours a larger number of smaller well-separated clusters. These observations are evident by comparing the cluster indices of Table 3.4 with Table 3.5.

Although the cohesion factor f_c appears to be like the c initial cluster prescription of FCM or KNN, it is functionally quite different. The cohesion factor does not find the num-

ber of clusters prescribed by c but instead, determines the number of points in a cluster and thus its maximum radius. Moreover it is observed experimentally that the cluster result of the cohesion factor possesses properties of scale, rotational and translational invariance, provided the data set contains structurally similar clusters. These properties are evident by comparing the features of the cluster radii and centres of Figs. 3.4 and 3.5 with Figs. 3.6 and 3.7, for the same cohesion factors. Both Figs. 3.6 and 3.7 are derived from the same data set as Figs. 3.4 and 3.5, except for a transposition of axes and the removal of one intermediate cluster (to simulate the appearance of different clusters). The data set of Figs. 3.6 and 3.7, called the *modified* Ruspini's data (derived from Fig. 3.4), refers to these changes.

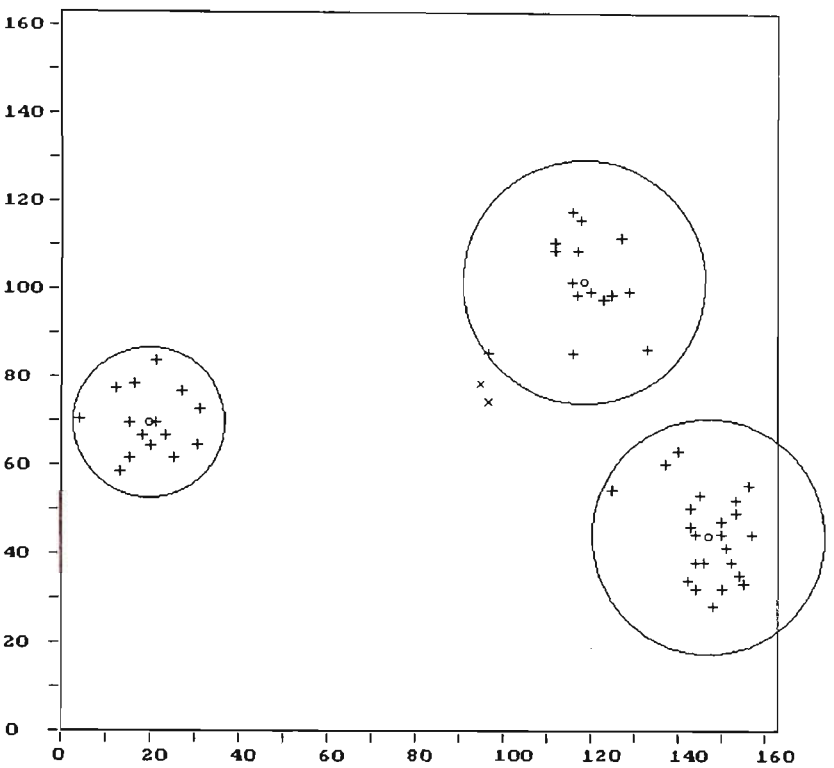


Figure 3.6 Three clusters detected from the *modified* data set of Fig. 3.4, using variable η_j. The cohesion factor, $f_c = 9$. Minimum cluster radius, $r_{min} = 17$.

Cluster	Centre (pixel units)	Radius (pixel units)	Cluster index, $\bar{s}_1$	N_α
1	19.4, 68.93	17.00	0.9129	15
2	146.04, 43.91	26.46	0.8267	23
3	117.53, 101.13	27.46	0.8201	15

Table 3.6 Summary of cluster statistics from the data set of Fig. 3.6, for a cohesion factor $f_c = 9$ and $r_{min} = 17$. Data set cluster index $\bar{s}_2 = 0.8492$. N_α is the number of points in the cluster.

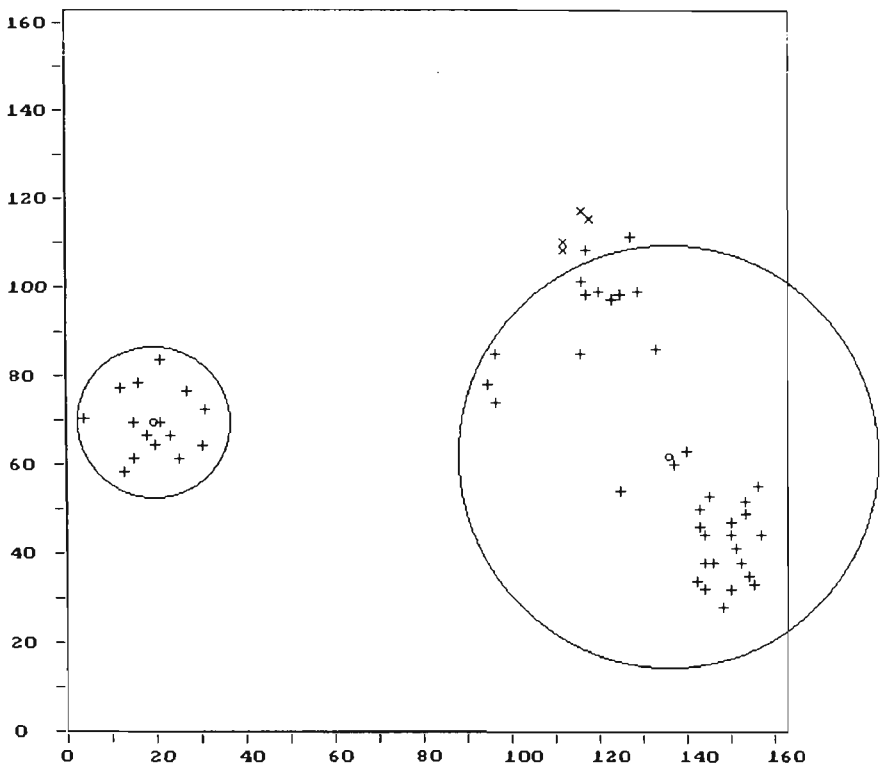


Figure 3.7 Two clusters detected from the data set of Fig. 3.6, using variable η_j. The cohesion factor, f_c = 17. Minimum cluster radius, r_{min} = 17.

Cluster	Centre (pixel units)	Radius (pixel units)	Cluster index, $\bar{s}_1$	N_α
1	19.4, 68.93	17.00	0.9226	15
2	134.94, 61.92	47.46	0.7404	36

Table 3.7 Summary of cluster statistics from the data set of Fig. 3.6, for a cohesion factor f_c = 17 and r_{min} = 17. Data set cluster index $\bar{s}_2$ = 0.7940. N_α is the number of points in the cluster.

The variable η_j feature in CPCM enhances the generalisation capability of an algorithm to detect clusters having regional shapes other than circular. To a limited extent, this is also possible with the fixed cluster radius of the CPCM algorithm, although the clusters detected from the latter is more restricted and in this sense less general than the case of variable η_j. Two interesting properties relating to the cohesion factor are obtained from the variable η_j algorithm. A bigger cohesion factor f_c will detect larger cluster substructures. If the cohesion factor f_c is made sufficiently small (eg. $0 < f_c < 1$), then the variable η_j algorithm reverts to the fixed η or fixed radius algorithm, because all clusters are limited to r_{min}. Therefore, the variable η_j algorithm is a generalisation of the fixed radius CPCM clustering algorithm.

3.4.2 Elliptic Cluster Structure

Ellipses are another common cluster structure in pattern recognition and also quite useful for practical applications. In the case of an elliptic region, we use a modified distance mesure of Gustafson-Kessel's Eq. (2.4.33) given by

$$d_{k,j}^2 = (\mathbf{x}_k - \mathbf{v}_j)^T F_j^{-1} (\mathbf{x}_k - \mathbf{v}_j) \tag{3.23}$$

where d_{kj} represents the distance of a point $\mathbf{x}_k$ from its prototype $\mathbf{v}_j$ in the jth cluster. The fuzzy covariance matrix F has the same interpretation as (2.4.32), repeated below for the CPCM algorithm

$$F_j = \frac{\sum_{k=1}^{N_j} u_k^2 (\mathbf{x}_k - \mathbf{v}_j)(\mathbf{x}_k - \mathbf{v}_j)^T}{\sum_{k=1}^{N_j} u_k^2} \tag{3.24}$$

Membership in the jth cluster can take a variety of unconstrained forms reviewed in Section 2.4.7. For the ellipse detection algorithm, we used a possibilistic membership

$$u_k = \frac{1}{1 + d_k^2 / \eta_j^2} \tag{3.25}$$

Substituting (3.24) into (3.23), the metric for $\mathbf{x}_k$ and $\mathbf{v}_j$ in $\Re^2$ may be reformulated for the feature components as

$$d_k^2 = \frac{\left(d_{k,x}^2 C_{22} - 2 d_{k,x} d_{k,y} C_{12} + d_{k,y}^2 C_{11}\right) \sum_{k=1}^{N_j} u_k^2}{C_{11} C_{22} - C_{12}^2} \tag{3.26}$$

where

$$C_{11} = \sum_{k=1}^{N_j} u_k^2 d_{k,x}^2 \tag{3.27a}$$

$$C_{12} = \sum_{k=1}^{N_j} u_k^2 d_{k,x} d_{k,y} \tag{3.27b}$$

$$C_{22} = \sum_{k=1}^{N_j} u_k^2 d_{k,y}^2 \tag{3.27c}$$

where $d_{k,x}$ and $d_{k,y}$ are the respective x and y coordinate components of the relative vector, $(\mathbf{x}_k - \mathbf{v}_j)$. The terms C_{11}, C_{12} and C_{22} are the coefficients of the 2 × 2 positive definite fuzzy covariance matrix F. The η_j factor is calculated as

$$\eta_j^2 = \frac{\sum_{k=1}^{N_j} \|\mathbf{x}_k - \mathbf{v}_j\|^2}{N_j} \tag{3.28}$$

and the prototype update has two forms given by

$$\mathbf{v}_j = \frac{\sum_{k=1}^{N_j} u_k^2 \mathbf{x}_k}{\sum_{k=1}^{N_j} u_k^2} \tag{3.29a}$$

$$\mathbf{v}'_j = \frac{\sum_{k=1}^{N_j} \mathbf{x}_k F_k}{\sum_{k=1}^{N_j} F_k} \tag{3.29b}$$

$$F_k = \begin{cases} 1 & \text{if} \quad u_k > \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

A CPCM algorithm to detect elliptic clusters.

Fix α_t, q and N_{min}.

Repeat

Assume $C_{11} = C_{12} = C_{22} = 0$, assign data centroid to $\mathbf{v}_0$ and randomise U_0.

Repeat

Calculate η_j from (3.28).

Calculate $\mathbf{v}_j$ from (3.29a).

Calculate d_k from (3.26) only if $C_{11}C_{22} \neq C_{12}^2$.

Calculate u_k from (3.25).

Until $\|\mathbf{v}_{j,t} - \mathbf{v}_{j,t-1}\| < \delta$.

Assign nearest neighbour of prototype $\mathbf{v}$ to $\mathbf{v}$.

Repeat

Calculate η_j from (3.28).

Calculate d_k from (3.26).

Calculate u_k from (3.25).

Calculate $\mathbf{v}'_j$ from (3.29b).

Until $\|\mathbf{v}'_{j,t} - \mathbf{v}'_{j,t-1}\| < \varepsilon$.

If $(0 \leq N_\alpha < N_{min})$ **Then** Remove N_α and update N_c.

If ($N_\alpha \geq N_{min}$) **Then** Save and remove cluster points of prototype **v** and update N_c.

Until ($N_c < N_{min}$).

Note: δ and ε are small values to control the stopping point, and t is the iteration index.

The two ellipses detected by this algorithm are shown in Fig. 3.8, for the same data set as Figs. 3.3. Ellipse statistics are given in Table 3.8. The ellipses are delineated by boundaries drawn at $\alpha_t = 0.9$.

Cluster	Centre	N_α
1	9.00, 6.54	13
2	10.27, 12.00	11

Table 3.8 Summary of cluster statistics from the elliptic cluster algorithm, for the data set of Fig. 3.3. Cluster parameters are $\alpha_t = 0.91$ and $N_{min} = 5$. N_α is the number of points in the cluster

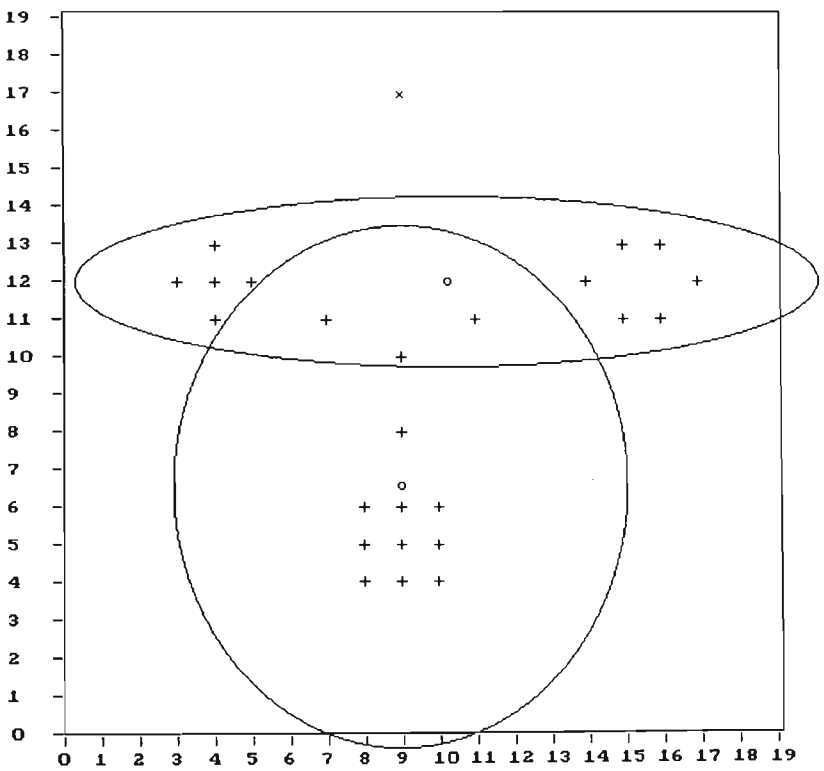


Figure 3.8 Two elliptic clusters detected from a data set of 25 points (modified from Kaufman and Rousseeuw's data set of Fig. 2.11). Cluster parameters are $\alpha = 0.91$ and $N_{min} = 5$. Note: "+" = cluster point and "×" = non-cluster point.

The ellipse algorithm that produced the result shown in Fig. 3.8 is quite sensitive to the alphacut α_t. In other words, good elliptic clusters are found only for a very narrow

range of α_t. In the next example, linear clusters are detected from the modified ellipse equations.

3.4.3 Linear Cluster Structure

To detect linear clusters, we intensify clustering of the elliptic region by recomputing the fuzzy covariance coefficients C_{11}, C_{12} and C_{22}. The algorithm is given below:

A CPCM algorithm to detect linear clusters.

Fix α_t, q and N_{min}.

Repeat

Assume $C_{11} = C_{12} = C_{22} = 0$, assign data centroid to $\mathbf{v}_0$ and randomise U_0.

Repeat

Calculate η_j from (3.28).

Calculate $\mathbf{v}_j$ from (3.29a).

Calculate d_k from (3.26) only if $C_{11}C_{22} \neq C_{12}^2$.

Calculate u_k from (3.25).

Until $\|\mathbf{v}_{j,t} - \mathbf{v}_{j,t-1}\| < \delta$.

Assign nearest neighbour of prototype $\mathbf{v}$ to $\mathbf{v}$.

Repeat

Zero the coefficients C_{11}, C_{12} and C_{22}.

Calculate $\mathbf{v}'_j$ from (3.29b).

Calculate η_j from (3.28).

Calculate d_k from (3.26) only if $C_{11}C_{22} \neq C_{12}^2$.

Calculate u_k from (3.25).

Until $\|\mathbf{v}_{j,t} - \mathbf{v}_{j,t-1}\| < \varepsilon$.

If $(0 \leq N_\alpha < N_{min})$ **Then** Remove N_α and update N_c.

If $(N_\alpha \geq N_{min})$ **Then** Save and remove cluster points of prototype $\mathbf{v}$ and update N_c.

Until $(N_c < N_{min})$.

Note: δ and ε are small values to control the stopping point, and t is the iteration index.

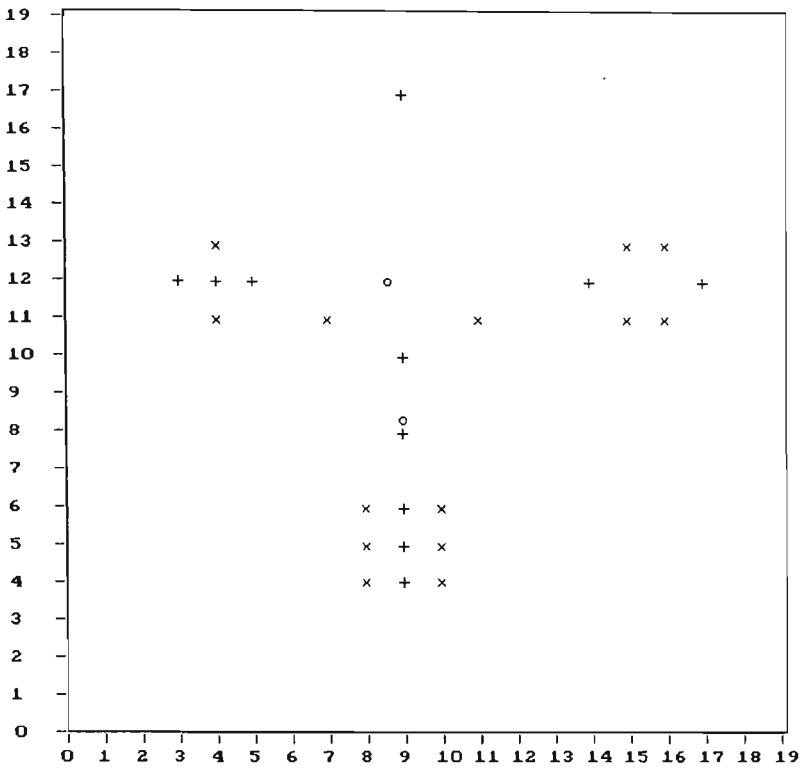


Figure 3.9 Two linear clusters detected from a data set of 25 points (modified from Kaufman and Rousseeuw's data set of Fig. 2.11). Cluster parameters are $\alpha_t = 0.95$ and $N_{min} = 5$. Note: "+" = cluster point and "×" = non-cluster point.

Like the algorithm of Fig. 3.8, the linear clustering algorithm is also quite sensitive to the α_t cluster parameter. The cluster statistics are summarised in Table 3.9.

Cluster	Centre	N_α
1 (vertical orientation)	9.00, 8.33	6
2 (horizontal orientation)	8.60, 12.00	5

Table 3.9 Summary of cluster statistics from the linear cluster algorithm, for the data set of Fig. 3.3. Cluster parameters are $\alpha_t = 0.95$ and $N_{min} = 5$. N_α is the number of points in the cluster.

3.4.4 Ring-Shape Cluster Structure

The ring-shape clustering algorithm presented here is fairly basic, but is adequate to demonstrate the feasibility of adapting conventional fuzzy clustering equations to the CPCM framework. It detects ring-shape clusters satisfactorily but needs more refinements to suit practical applications. Practical issues are considered more fully in Chapter 6.

Equations for the ring-shape cluster algorithm are nearly identical to the round region cluster algorithm. The cluster to be detected is assumed to have a fixed radius (other forms are possible). Consequently, a simple solution is given by the metric

$$d_k = \|\mathbf{x}_k - \mathbf{v}_j\| - r_j \tag{3.30}$$

where r_j is the radius of the jth cluster. The membership is defined as

$$u_k = \frac{1}{\exp(|d_k|/\eta_j)} \tag{3.31}$$

where

$$\eta_j = s\sqrt{\frac{\sum_{k=1}^{N_j}\|\mathbf{x}_k - \mathbf{v}_j\|^2}{N_j}} \tag{3.32}$$

A solution to the metric of (3.30) will maximise the membership of (3.31), and thus minimise the objective function. This occurs when $r_j = \|\mathbf{x}_k - \mathbf{v}_j\|$. The algorithm is given below:

A CPCM algorithm to detect ring-shape clusters.

Fix r_j, α_t, s and N_{min}.

Repeat

Assume initial $\mathbf{v}_0$ as nearest neighbour of data centroid.

Repeat

Calculate η_j from (3.32).

Calculate d_k from (3.30).

Calculate u_k from (3.31).

Calculate $\mathbf{v}_j$ from (3.29a).

Until $\|\mathbf{v}_{j,t} - \mathbf{v}_{i,t-1}\| < \delta$.

Assign nearest neighbour of prototype $\mathbf{v}$ to $\mathbf{v}$.

Repeat

Calculate η_j from (3.32).

Calculate d_k from (3.30).

Calculate u_k from (3.31).

Calculate $\mathbf{v}'_j$ from (3.29b).

Until $\|\mathbf{v}'_{j,t} - \mathbf{v}'_{j,t-1}\| < \varepsilon$.

If $(0 \le N_\alpha < N_{min})$ **Then** Remove N_α and update N_c.

If $(N_\alpha \ge N_{min})$ **Then** Save and remove cluster points of prototype $\mathbf{v}$ and update N_c.

Until $(N_c < N_{min})$.

Note: δ and ε are small values to control the stopping point and t is an iteration index.

Figure 3.10 shows a ring-shape cluster detected from the data set. The selected cluster parameters $r_j = 4$, $\alpha_t = 0.95$, $s = 1$ and $N_{min} = 10$, produced a cluster centred at (4.77, 5.23) with a radius of 4.25 units, indicated by the circular boundary. The true centres are at (5, 5) and (9, 9) and the true radius for each cluster is 4 units. The algorithm seems to detect only a single cluster at a time. Another cluster centred at (9, 9), was not detected. However, selecting cluster parameters $\alpha_t = 0.95$ and scale factor $s = 2$ successfully detect the second cluster centred at (9.21, 9.21), yielding a cluster radius of 4.27 units.

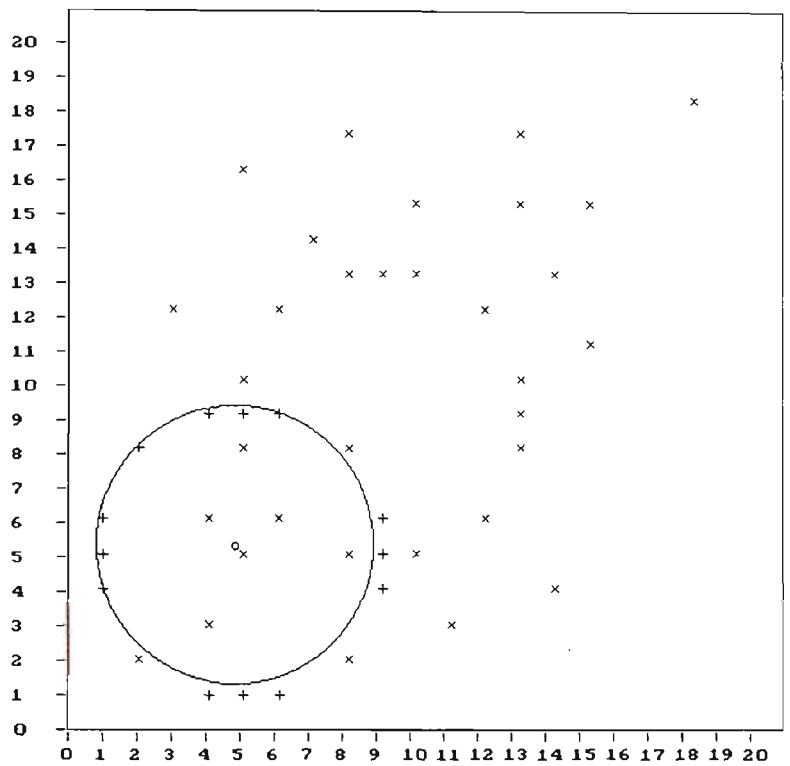


Figure 3.10 One ring-shape cluster detected from the data set. The selected cluster parameters were $\alpha_t = 0.95$, $s = 1$ and $N_{min} = 10$. Note: "+" = cluster point and "×" = non-cluster point.

3.5 Clustering Performance

We used Ruspini's data set (see Fig. 3.4) to compare clustering performance. Our data set of Fig. 3.4 agrees with Kaufmann and Rousseeuw's tabulation of Ruspini's data [Kaufman and Rousseeuw, 1990, p. 100]. This version differs from [Diday and Simon, 1976, p.72] by a transposition of coordinates. We evaluate the clustering performances of three algo-

rithms: (i) KNN, (ii) CPCM variable η_j for round region and (iii) FCM, and examine the performance criteria in terms of convergence characteristics and cluster validity based on (3.14), for several clusters of Ruspini's data set.

3.5.1 Convergence Characteristics

The convergence characteristics for four clusters are shown in Figs. 3.11 and 3.12 (x, y-coordinates of cluster number *1*), Figs. 3.13 and 3.14 (x, y-coordinates of cluster number *2*), Figs. 3.15 and 3.16 (x, y-coordinates of cluster number *3*) and Figs. 3.17 and 3.18 (x, y-coordinates of cluster number *4*). From these results, the following observations are made:

- Both the KNN and the CPCM algorithms converge quite rapidly compared to FCM.
- The convergence rate of CPCM is like that of KNN.
- Both the KNN and the CPCM algorithms converge to the same local centroids. FCM does not converge to local cluster centroids (for fixed $m = 2$, $p = 2$).

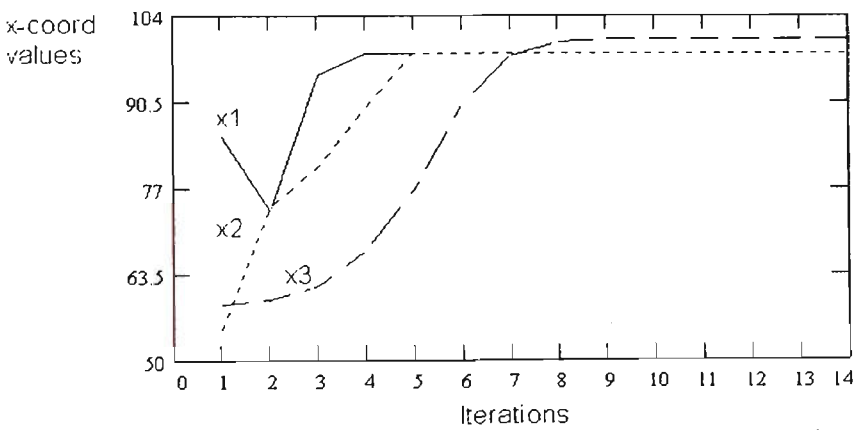


Figure 3.11 Prototype convergence of number *one* of four clusters. The three curves represent the x coordinate component from Ruspini's data set of Fig. 3.4. The three algorithms corresponding to symbols x1 = KNN, x2 = CPCM round region with variable η_j, and x3 = FCM converge to 98.2, 98.2 and 100.4 respectively.

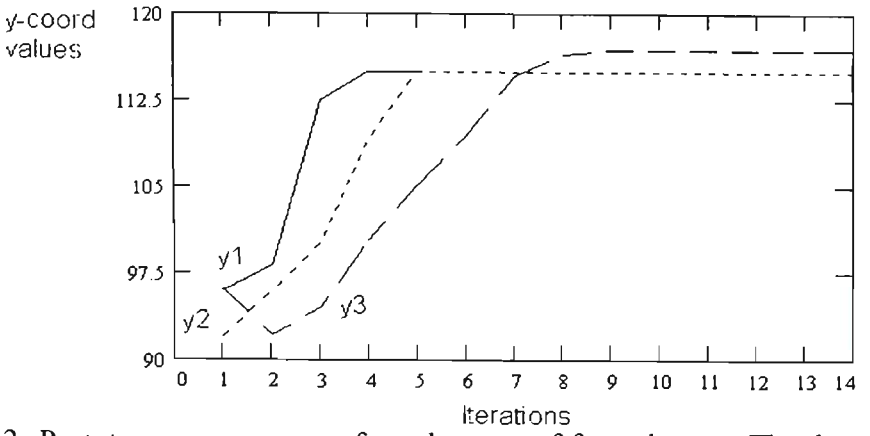


Figure 3.12 Prototype convergence of number *one* of four clusters. The three curves represent the *y* coordinate component from Ruspini's data set of Fig. 3.4. The three algorithms corresponding to symbols y1 = KNN, y2 = CPCM round region with variable η_j, and y3 = FCM converge to 114.9, 114.9 and 116.8 respectively.

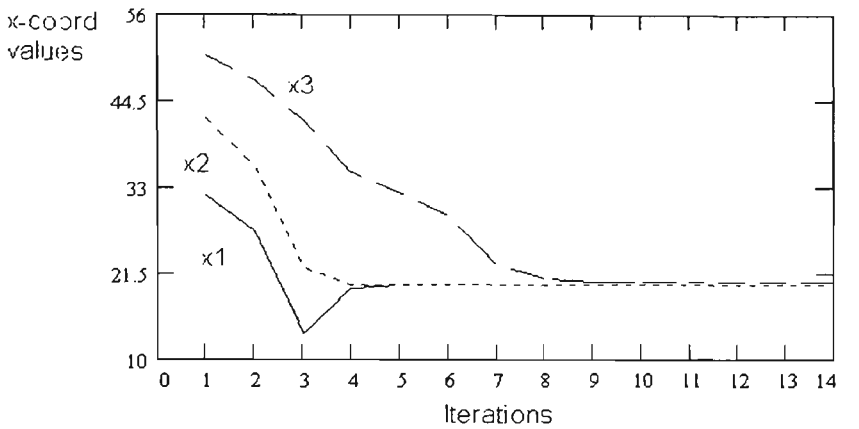


Figure 3.13 Prototype convergence of number *two* of four clusters. The three curves represent the *x* coordinate component from Ruspini's data set of Fig. 3.4. The three algorithms corresponding to symbols x1 = KNN, x2 = CPCM round region with variable η_j, and x3 = FCM converge to 20.1, 20.1 and 20.5 respectively.

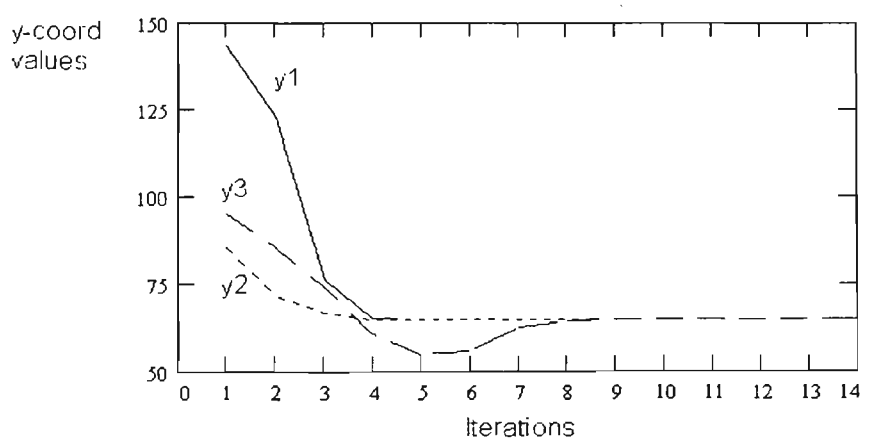


Figure 3.14 Prototype convergence of number *two* of four clusters. The three curves represent the *y* coordinate component from Ruspini's data set of Fig. 3.4. The three algorithms corresponding to symbols y1 = KNN, y2 = CPCM round region with variable η_j, and y3 = FCM converge to 64.9, 64.9 and 64.9 respectively.

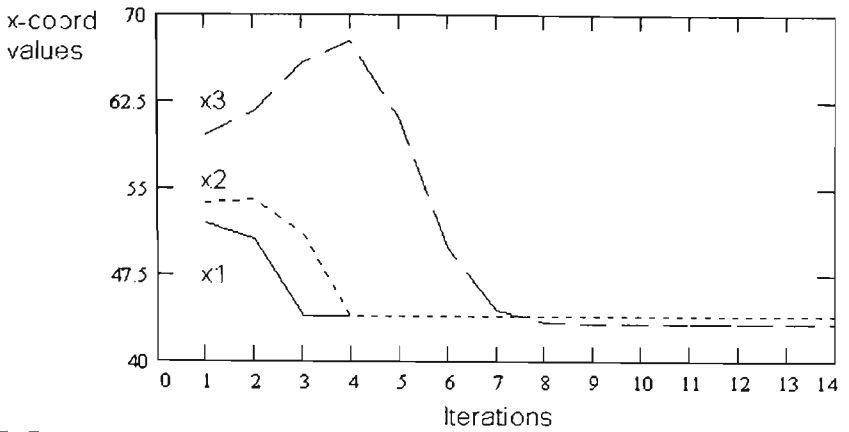


Figure 3.15 Prototype convergence of number *three* of four clusters. The three curves represent the x coordinate component from Ruspini's data set of Fig. 3.4. The three algorithms corresponding to symbols x1 = KNN, x2 = CPCM round region with variable η_j, and x3 = FCM converge to 43.9, 43.9 and 43.3 respectively.

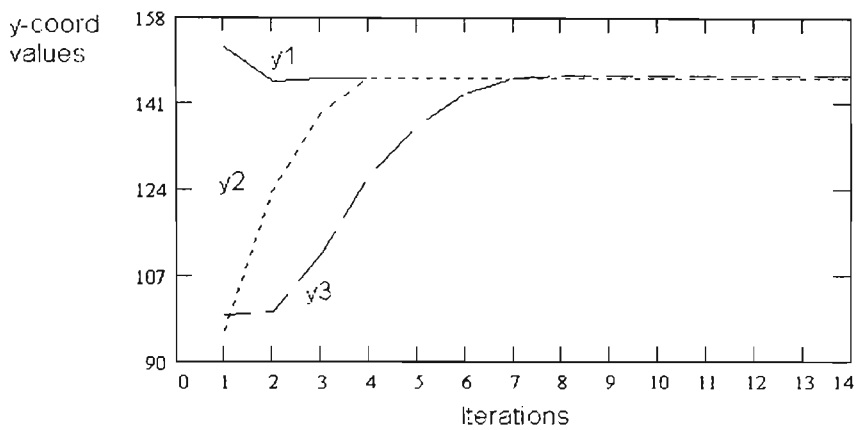


Figure 3.16 Prototype convergence of number *three* of four clusters. The three curves represent the y coordinate component from Ruspini's data set of Fig. 3.4. The three algorithms corresponding to symbols y1 = KNN, y2 = CPCM round region with variable η_j, and y3 = FCM converge to 146, 146 and 146.5 respectively.

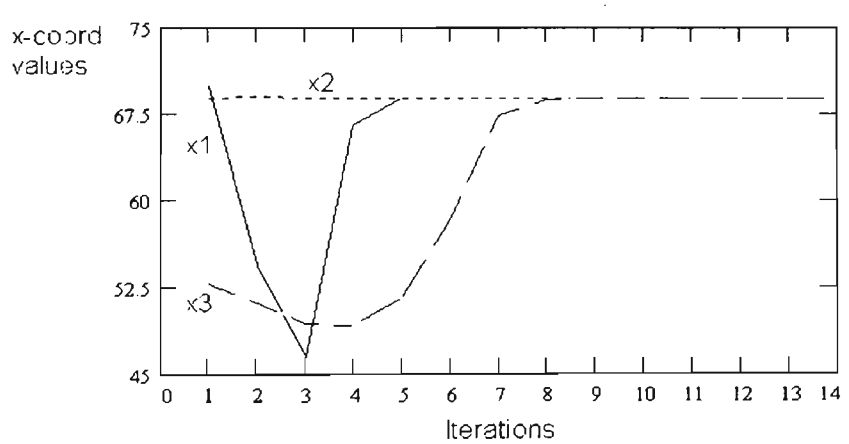


Figure 3.17 Prototype convergence of number *four* of four clusters. The three curves represent the x coordinate component from Ruspini's data set of Fig. 3.4. The three algorithms corresponding to symbols x1 = KNN, x2 = CPCM round region with variable η_j, and x3 = FCM converge to 68.9, 68.9 and 68.8 respectively.

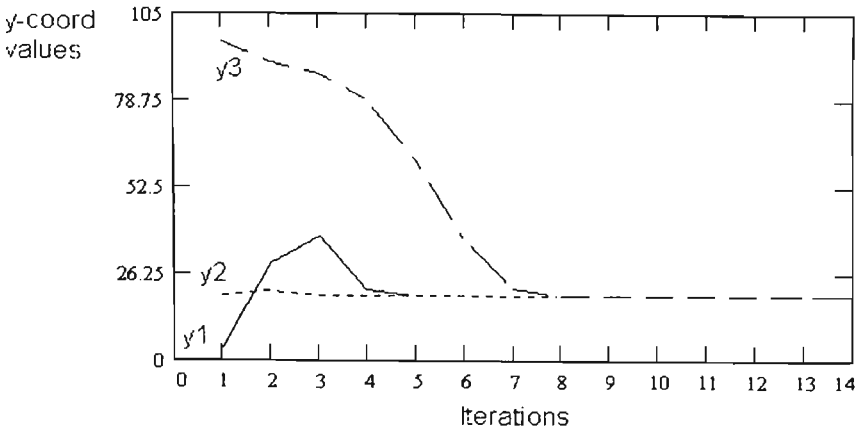


Figure 3.18 Prototype convergence of number *four* of four clusters. The three curves represent the y coordinate component from Ruspini's data set of Fig. 3.4. The three algorithms corresponding to symbols y1 = KNN, y2 = CPCM round region with variable η_j, and y3 = FCM converge to 19.4, 19.4 and 19.7 respectively.

For $c = 4$, both FCM and CPCM algorithms produce repeatable cluster results. The KNN algorithm was sensitive to initial random neighbour points (starting point) and converges to different cluster centres (termination point) as shown in Table 3.10.

Trial number	Cluster number	Starting point		Termination point	
		x-coord	y-coord	x-coord	y-coord
1	1	85	96	98.2	114.9
	2	32	143	20.1	64.9
	3	52	152	43.9	146.0
	4	70	4	68.9	19.4
2	1	85	96	98.2	114.9
	2	98	116	56.0	137.0
	3	78	94	40.6	148.6
	4	10	59	41.1	45.4
3	1	85	115	98.2	114.9
	2	32	143	52.1	143.5
	3	28	147	36.4	148.4
	4	61	25	41.1	45.4

Table 3.10 Summary of the effects of random initial neighbour points on prototype positions of the KNN algorithm, for four prescribed clusters from the data set of Fig. 3.4.

For $c \neq 4$, both FCM and KNN algorithms were sensitive to initial starting points. In the case of FCM, random initial memberships were used to determine the initial prototypes. Table 3.11 shows the effects of random initial memberships on FCM prototypes, for $c = 3$. The resulting prototype positions in each case were different. Trials on five clusters also show variations in the prototype results. The reason that repeatable results were obtained

from FCM for $c = 4$ is attributed to the well separated and compact nature of four clusters. For $c = 3$ or $c = 5$, the cluster boundaries are closer together also increase the probability of forming different clusters, and thus the nonrepeatable results.

Trial number	Cluster number	Starting point		Termination point	
		x-coord	y-coord	x-coord	y-coord
1	1	52.70	100.44	59.80	137.88
	2	59.83	88.85	69.67	22.66
	3	52.15	94.61	23.18	67.09
2	1	57.21	91.61	98.08	114.27
	2	62.20	92.20	42.87	144.47
	3	56.06	90.01	43.00	13.13

Table 3.11 Summary of the effects of random initial memberships on prototype positions of the FCM algorithm, for $c = 3$ from the data set of Fig. 3.4.

The cluster result of CPCM, like the FCM and KNN algorithms, is also affected by initial prototype position. In the case of CPCM, there is a consistent basis to initial prototype estimation (because of data centroid assumption). However, unlike both FCM and KNN algorithms, the same cluster result (eg. cluster index, number of clusters, cluster radius and prototype) can be produced by adjusting the cohesion factor f_c. Table 3.12 shows the f_c values needed to produce the same cluster result (four complete clusters at $r_{min} = 17$) at three different initial prototype positions compared to the algorithm's normal initial prototype positions. For example, start position (10,80) for $f_c = 9$ gives the same cluster result as the normal (assumed) position (54.9,92) with $f_c = 11$. The CPCM algorithm can also optimise the cluster index by adjusting f_c and r_{min}. This feature, unavailable in either the FCM or KNN algorithms, helps to identify possible noise points and improves the cluster structure. The next section elaborates on this feature.

	Starting position (10, 80)	Normal position (54.9, 92)	Starting position (70, 20)	Normal position (54.9, 92)	Starting position (110, 120)	Normal position (54.9, 92)
f_c	9	11	9	13	9	11

Table 3.12 Several values of f_c that produce equivalent cluster result of four complete clusters at $r_{min} = 17$, from the data set of Fig. 3.4.

3.5.2 Comparing the Cluster Index

The data set cluster index defined by $\bar{s}_2$ as Eq. (3.14), is useful in determining the optimal size of the cluster numbers. The criteria of Eqs. (3.12a), (3.12b) and (3.12c) favours a clustering result that has compact cluster regions and good separation between cluster prototypes. One significant advantage of this criteria for defining cluster validity is its simplicity and effectiveness, involving only simple operations like summing, division and subtraction.

Number of clusters	Data set cluster index, $\bar{s}_2$		
	KNN	CPCM	FCM
2	0.6514	0.6154	0.6464
3	0.7269	0.7269	0.7229
4	0.8158	0.8158	0.8181
5	0.7467	0.7891	0.7924

Table 3.13 A comparison of the data set cluster index from KNN, CPCM and FCM algorithms for a range of cluster numbers from the data set of Fig. 3.4. Maximum index is shown shaded.

Table 3.13 compares the data set cluster index from each of the three algorithms, for several clusters from Ruspini's data set (see Fig. 3.4). In each case, the index peaked at four clusters. This result agrees with the analyses of [Diday and Simons, 1976, p.71] and also [Kaufman and Rousseeuw, 1990, p.101]. Figure 3.4 indicates that four clusters give the most natural interpretation.

It is interesting to note that FCM gives the best clustering performance with a data similarity index of 0.8181 for four clusters, but has a limited optimisation scope. FCM's result may be attributed to the LSE criterion of the objective function and the choice of $m = 2$. CPCM has a greater optimisation scope, by allowing selective exclusion of cluster points, to maximise the data set similarity index from a particular selection of f_c and r_{min} parameters. Table 3.14 reveals a higher index for CPCM at $\bar{s}_2 = 0.8568$, compared to the fully clustered case of Fig. 3.4 where $\bar{s}_2 = 0.8158$. The cluster substructures corresponding to $\bar{s}_2 = 0.8586$, shown on Fig. 3.19, are more compact compared to those of Fig. 3.4.

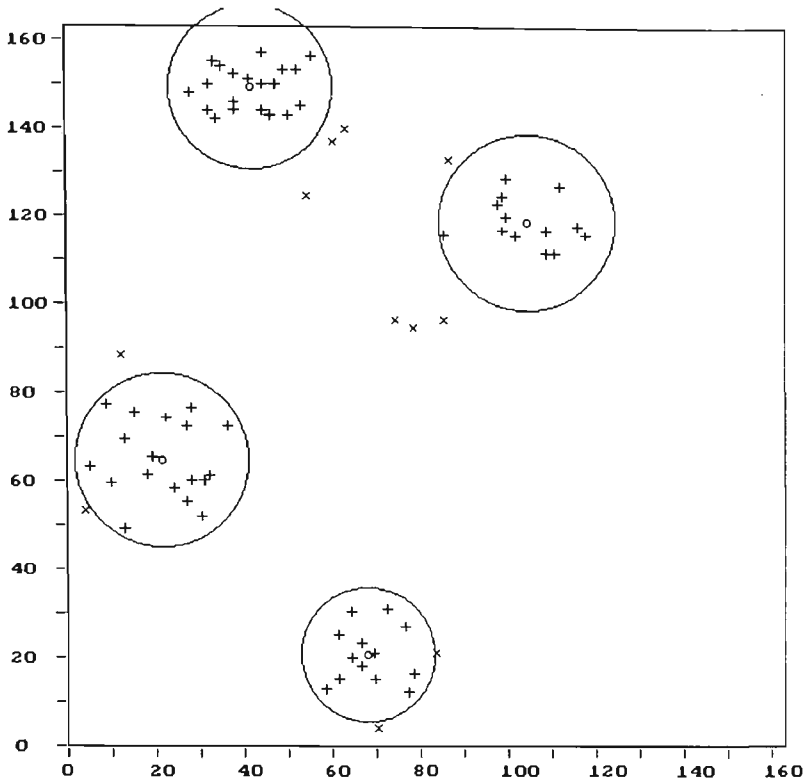


Figure 3.19 Cluster result from CPCM algorithm for $\bar{s}_2 = 0.8586$.

$\bar{s}_2$	f_c	r_{min}	N_α	Prototype	N_u	Unclustered points
0.8158	9	17	17	98.2,114.9	0	Nil
			20	20.1,64.9		
			23	43.9,146.0		
			15	68.9,19.4		
0.8492	7	17	19	20.6,63.7	7	12,88; 54,124; 86,132; 85,115; 85,96 78,94; 74,96
			22	43.4,147.0		
			15	68.9,19.4		
			12	105.1,118.3		
0.8571	6	17	18	21.5,64.3	9	4,53; 12;88; 54,124; 60,136; 63,139 86,132; 85,96; 78,94; 74,96
			13	103.5,118.1		
			20	41.6,148.0		
			15	68.9,19.4		
0.8586	6	15	18	21.5,64.3	11	4,53; 12,88; 54,124; 60,136; 63,139 86,132; 85,96; 78,94; 74,96; 70,4 83,21
			13	103.5,118.1		
			20	41.6,148.0		
			13	67.8,20.5		

Table 3.14 Summary of data set cluster indices obtained from the data set of Fig. 3.4 under different conditions of the cohesion factor f_c and the minimum cluster radius r_{min} of the CPCM algorithm. The symbols N_α and N_u represent the number of points in the cluster, and the number of unclustered points, respectively. The similarity index for the data set is denoted by $\bar{s}_2$. Note: maximum similarity index has a value of 0.8586. Maximum index is shown shaded.

There are two points to note about the cluster result of Section 3.5.2. Firstly, optimum cluster index does not imply optimum cluster structure, although it does in most cases. Secondly, the optimum index obtained by optimising parameters f_c and r_{min} has to be weighed against the significance of the cluster result. Ignoring a few cluster points at the fringe or shrinking the cluster regions will improve the cluster index, but taking this approach to extremity will result in loss of data structure.

Chapter 4

Segmentation of Regions

This chapter examines the practical applications relating to the detection of (i) wool contaminants and (ii) tile surface defects. Many conventional techniques of image processing exist for region segmentation but possibly only a few are suitable for the above applications. Methods that are suitable for this purpose are various types of thresholding such as peak-valley mode detection, adaptive thresholding and local area thresholding [Davies, 1990], boundary representational schemes such as chain coding and polygonal approximation, and Fourier descriptors [Gonzalez and Woods, 1992]. All these methods depend rather critically on an adequate lighting environment and illumination condition. For example, the peak-valley mode thresholding method requires a good definition of a valley between two peaks, a condition that rarely exists normally. Consequently, it is practically a necessity to induce a high contrast between object and background by controlling the lighting environment. A controlled lighting environment can incur substantial costs depending on the intensity of the illumination and the degree of control needed. For this reason, any alternative method that mitigates the need for an expensive lighting system will be considered beneficial.

The proposed fuzzy clustering algorithm to detect defect patterns is tolerant to some variation in illumination. To simplify the computation task, the images of both applications are processed as 256 gray levels in 256×256 resolution. Region segmentation is performed using the CPCM algorithm to detect round structures with fixed cluster parameters α_t, q and η. Clustering performance in terms of cluster structures and processing time are compared with the FCM algorithm. Following the introduction in Section 4.1, the segmen-

tation algorithm is reviewed in Section 4.2. Sections 4.3 presents the experimental result and Section 4.4 the conclusions.

4.1 Introduction

Wool contaminants refer to three types of vegetable matter, typically consisting of burrs (about 3 mm or greater), grass (about 2 mm) and dirt particles (about 1 to 2 mm). An example is illustrated in Fig. 4.1(a). These contaminants are normally removed by mechanical means such as a combing machine, by adjusting the size of the comb and the rate of combing. Current commercial vision systems using strip lighting to detect the material density are very costly and do not provide sufficient accuracy. The most reliable method is the standard industry method, using trained human inspectors to manually count and grade the contaminants. This is also costly and prone to human errors. The proposed method is simple and cost-effective. It assumes the wool is uniformly combed and sufficiently thin to enable the detection of contaminants by back lighting.

Tile surface defects refer to chipped edges or cracks, uneven surface texture or color variations. For the purpose of this thesis, only the problem of uneven tile surface due to inconsistent paint thickness is examined because this problem can be solved by a similar fuzzy clustering method. To enable detection of the defect patterns, the angle of incidence of the light on the tile's surface is adjusted to reflect sufficient light around the edge of paint flow (the defect pattern). An example is illustrated in Fig. 4.4(a).

4.2 Segmentation Algorithm

The segmentation algorithm uses the same CPCM framework described in Section 3.2. The membership equation is defined by

$$u_k = \frac{1}{\exp\left[d_\eta^q / \eta\right]} \tag{4.1}$$

where the fixed cluster radius d_η is defined by the fixed alphacut in

$$\eta = -\frac{d_\eta}{\ln \alpha_t} \tag{4.2}$$

and $d_\eta = |x_k - v_j|$. The prototype equation is defined by

$$v_j = \frac{\sum_{k=1}^{N_j} x_k F_k}{\sum_{k=1}^{N_j} F_k} \tag{4.3}$$

$$F_k = \begin{cases} 1 & \text{if} \quad u_k > \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

where v_j represents the prototype (intensity) of the jth cluster, x_k denotes the intensity value at the kth point position, and the fuzzifier exponent q is assumed equal to unity. The segmentation algorithm is given below:

A CPCM region segmentation algorithm.

Assume $\alpha_t = 0.9$, $q = 1$, $\eta = 400$ and $\varepsilon = 1$.

Repeat

Assume v_j from nearest neighbour of intensity centroid.

Repeat

Calculate u_k from (4.1).

Calculate v_j from (4.3).

Until $|v_{j,t} - v_{j,t-1}| < \varepsilon$.

If $(0 \le N_\alpha < N_{min})$ **Then** Remove N_α and update N_c.

If $(N_\alpha \ge N_{min})$ **Then** Save and remove cluster points of prototype **v** and update N_c.

Until $(N_c < N_{min})$.

Note: ε is a small value used to control the stopping point and t is an iteration index.

Prototypes obtained from this algorithm define the segmented regions according to the relation,

$$\text{Intensity range of region } j\text{: } 0 \le v_j \pm d_\eta \le 255 \tag{4.4}$$

where the cluster radius d_η is obtained from (4.2). Note that (4.4) assumes a separate procedure to check that the boundary limits of each prototype do not overlap. To use this algorithm correctly, the prototype is assumed to define the intensity centre of the defect pattern at the higher end of the intensity spectrum with the background at the lower end of the intensity spectrum. In other words, defect patterns are detected as brighter objects against a darker background. Consequently in the case of the wool image, the defect patterns which are acquired as dark spots in a light background (from back-lighting), are processed as an "inverted" or negative image. In the case of the tile image, image inversion is not required.

For these applications, $\eta = 400$ is satisfactory for the detection of small scale (< 3 % of data) defect patterns. If a large number of defect patterns are present in the image, this value of η tends to produce excessive residual noise (see Fig. 4.1(b)). To overcome this problem, an adjustment on the η value is made according to the formula

$$\eta^* v_h = 82500 \tag{4.5}$$

where η^* denotes the improved estimate for defect pattern identification and v_h is the highest prototype value from the prototype list. The constant of (4.5) is empirically determined for both applications. The following rules are suggested for the selection of η^*.

η selection rules (for the wool and tile applications).

Step 1. Initially assume $\eta = 400$ and generate a prototype list from the algorithm.

Step 2. Select the highest prototype value v_h from the prototype list.

Step 3. If the pixel count corresponding to v_h is less than 100, select the next highest v_h.

Step 4. Calculate η^* from (4.5) and generate a new prototype list from the algorithm.

Step 5. Select the highest prototype value v^* from the prototype list of η^*.

Step 6. If the pixel count of $v^* < 100$, use v_h from Step 2.

4.3 Experimental Results

4.3.1 Detection of Wool Contaminants

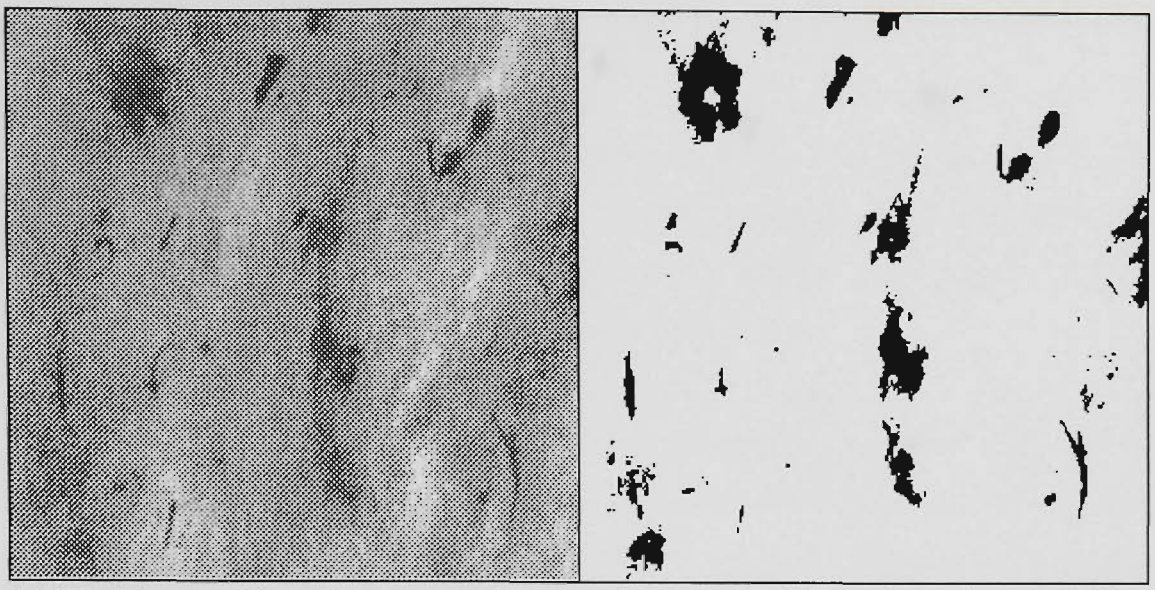
Two sets of results are presented from the wool problem. The first set, shown on Figs. 4.1(a) to 4.1(f), represents a moderate level of contamination. The other set, shown on Fig. 4.2(a) to 4.2(f), contains a lower level of contamination. This selection of contaminants is made to evaluate the effects of contaminant levels on the algorithm's performance.

Figure 4.1(a) of the first set shows the uncombed wool with different kinds of vegetable matter. A knot of entangled wool is visible at the upper left corner. The uneven wool texture is evident from the randomly distributed dark and bright spots. Vegetable matter appears as dark regions in the shape of small lines, arcs and spots. Figure 4.1(b) is the cluster result using $\eta = 400$. The segmented cluster contains residual noise that interferes with the identification of wool contaminants. Adjusting η according to the selection rules clearly improves the cluster result shown on Fig. 4.1(c). Applying blob analysis with an area threshold of 15 pixels (an area element counts as one pixel) removes unwanted resid-

ual noise from the segmented image of Fig. 4.1(c). The area threshold is a clustering criterion used in blob analysis to identify good blobs ie. blobs with areas greater than the threshold. The result shown on Fig. 4.1(d) gives a reasonable estimate of the contaminants identified from blob analysis. The small cross next to the blob identity number marks the location of the centroid of area of the blob. The area threshold provides a criterion for the removal of unwanted specks or residual noise from the segmented image of Fig. 4.1(c). Figure 4.1(e) shows the nearest FCM cluster to the defect pattern for $c = 10$. The FCM result of Fig. 4.1(e) agrees well with the CPCM result of Fig. 4.1(c).

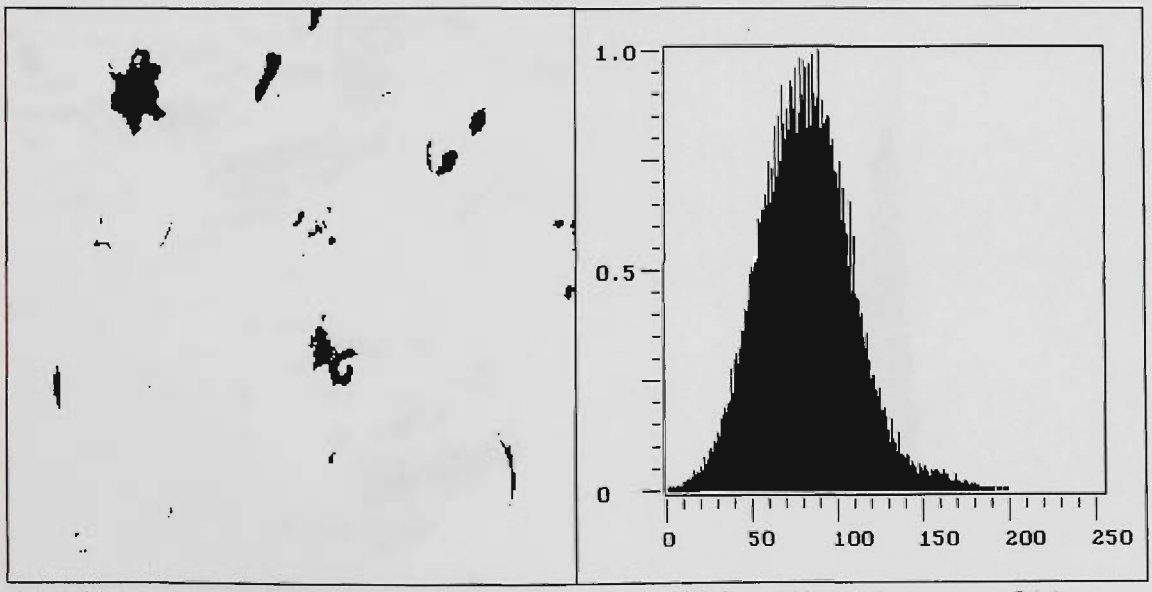
In the second set of results, Fig. 4.2(a) shows a reduced level of contaminants following a more extensive combing process. Segmentation of these particles, shown in Fig. 4.2(b), were obtained at $\eta^* = 439$. The result of blob analysis is shown in Fig. 4.2(d). Figure 4.2(c) shows unsatisfactory result from FCM at $c = 12$. The FCM cluster result at $c = 6$, in Fig. 4.2(e), is considerably worse. The extensive level of residual noise in both Figs. 4.2(c) and 4.2(e) exceed the capacity of blob analysis to estimate the contaminants present with reasonable accuracy. For this particular case, the numerous spurious blobs, especially the large one at the lower right corner, will be incorrectly detected as contaminants.

The histograms of the wool samples of Figs. 4.1(f) and 4.2(f) show no discernible valley to enable optimal thresholding. Moreover, the intensity boundary between the contaminants and wool material seems quite fuzzy. Consequently, methods that rely on detecting histogram profile characteristics cannot be expected to yield reliable results. Unlike the thresholding methods, the proposed CPCM region segmentation algorithm does not suffer from this problem.



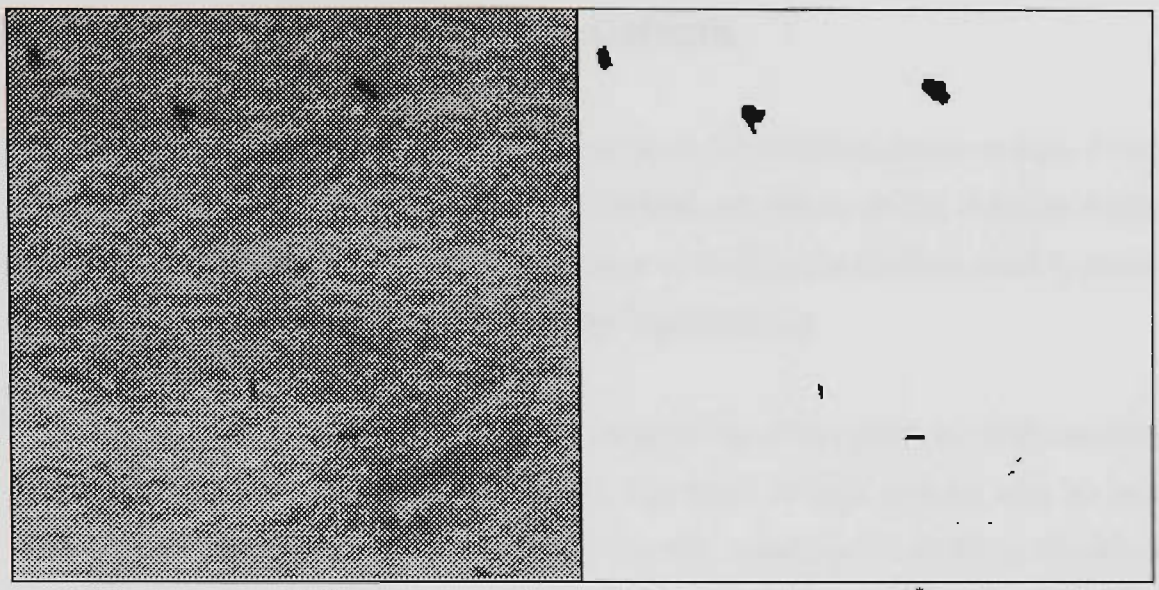
(a) Wool image with moderate contaminants. (b) Segmented image ($\eta = 400$, $\nu = 140$).

(c) Segmented image ($\eta^* = 589$, $\nu = 158$). (d) Blob analysis of (c) (area > 15 pixels).

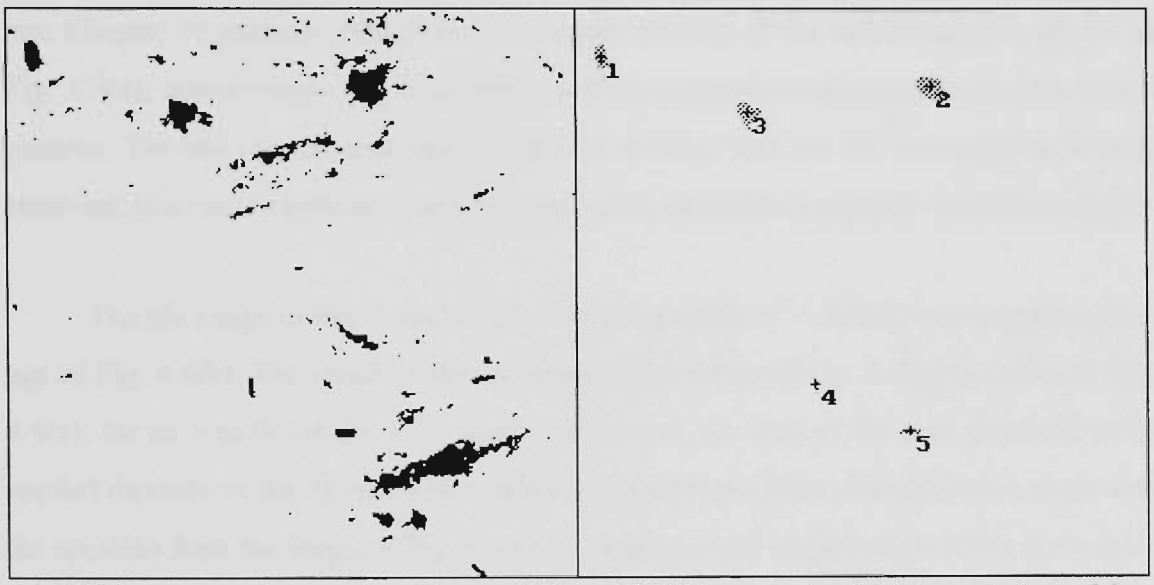


(e) FCM segmented image ($c = 10$, $\nu = 157$). (f) Normalised histogram of (a).

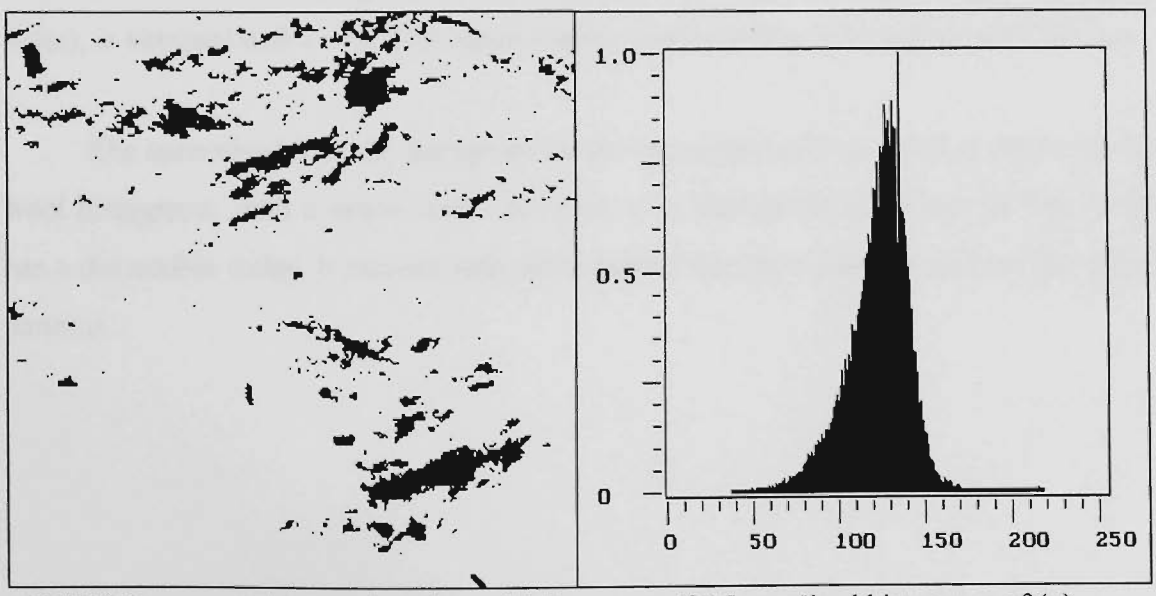
Figure 4.1 Wool with moderate level of contaminants.



(a) Wool image with few contaminants. (b) Segmented image ($\eta^* = 439$, $v = 193$).



(c) FCM segmented image ($c = 12$, $v = 155$). (d) Blob analysis of (b) (area > 5 pixels).



(e) FCM segmented image ($c = 6$, $v = 149$). (f) Normalised histogram of (a)

Figure 4.2 Wool with few contaminants.

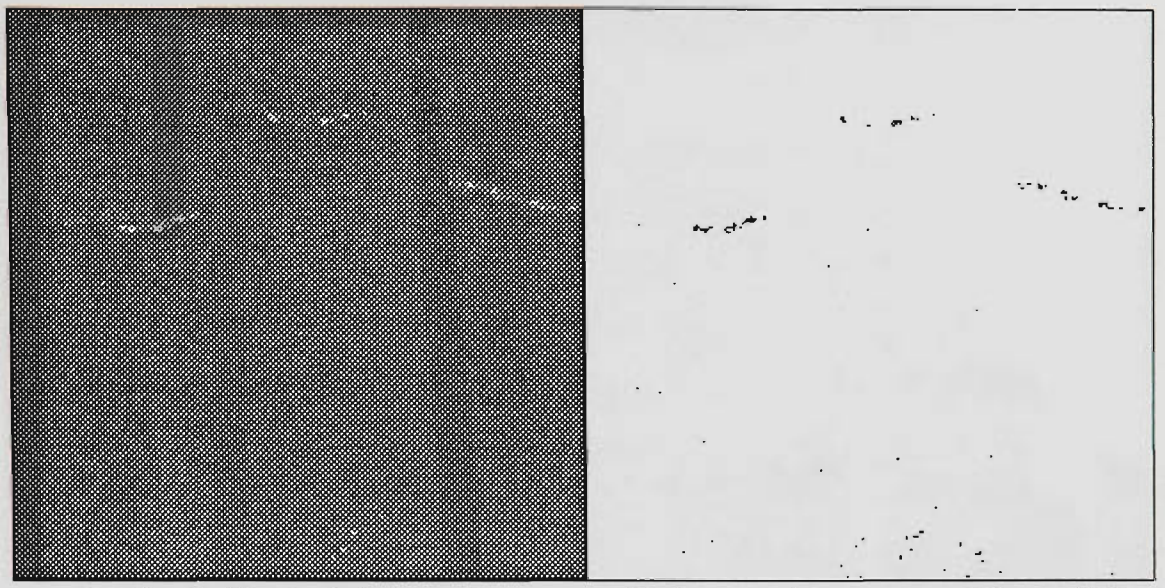
4.3.2 Detection of Tile Surface Defects

Two sets of results are presented from the tile problem. The first set, shown on Figs. 4.3(a) to 4.3(f), represents a low level illumination. The other set, shown on Fig. 4.4(a) to 4.4(f), contains a higher level illumination. This selection of illumination levels is made to evaluate effects of illumination levels on the algorithm's performance.

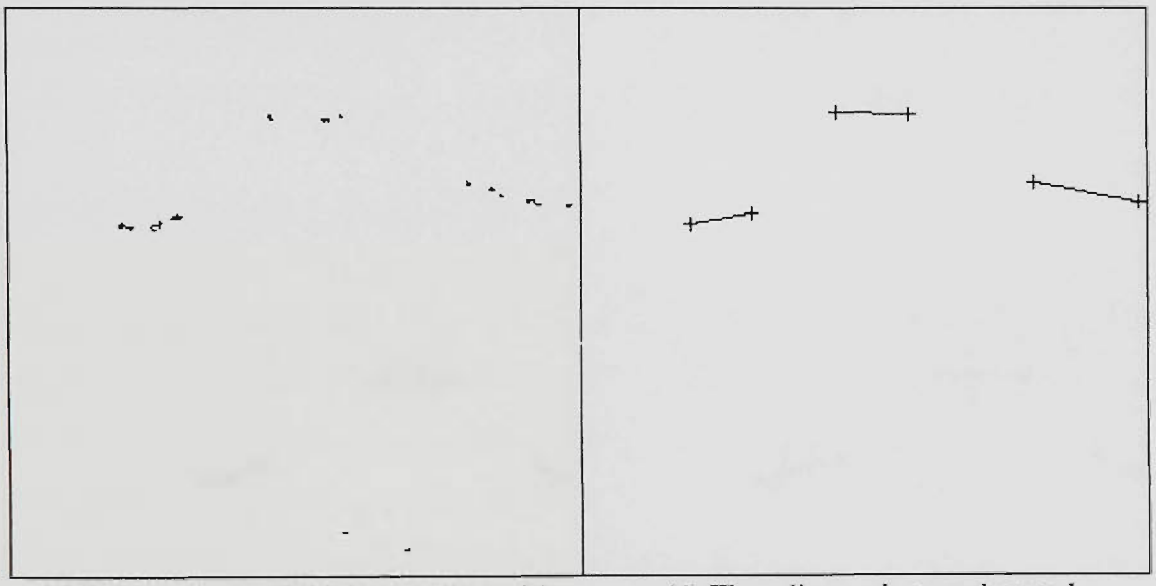
Applying the η selection rules, the tile image of Fig. 4.3(a) yields $\eta = 400$, resulting in the segmented image shown in Fig. 4.3(b). The result of blob analysis with an area threshold of 2 pixels, is shown in Fig. 4.3(c). For this particular tile problem, the defect patterns may be characterised by linear cluster approximations. A linear cluster algorithm (see Chapter 5) correctly detects the three main features of the defect patterns, shown in Fig. 4.3(d), demonstrating the effectiveness of the algorithm to characterise the tile defect patterns. The best cluster result from FCM was obtained with $c = 12$, shown in Fig. 4.3(e). However, this result contains excessive noise which cannot be corrected with blob analysis.

The tile image in Fig. 4.4(a) of the second set yields $\eta^* = 327$ for the segmented image of Fig. 4.4(b). The result of blob analysis of the image of Fig. 4.4(b) is shown in Fig. 4.4(c), for an area threshold of 50 pixels. In general, the level of the area threshold to be applied depends on the illumination condition of the image. Blob analysis removed most of the speckles from the image of Fig. 4.4(b) to enable correct identification of the three main defect pattern shown on Fig. 4.4(d). The optimal FCM cluster numbers, shown in Fig. 4.4(e), is obtained with $c = 6$. This result is quite similar to Fig. 4.4(b) of the CPCM case.

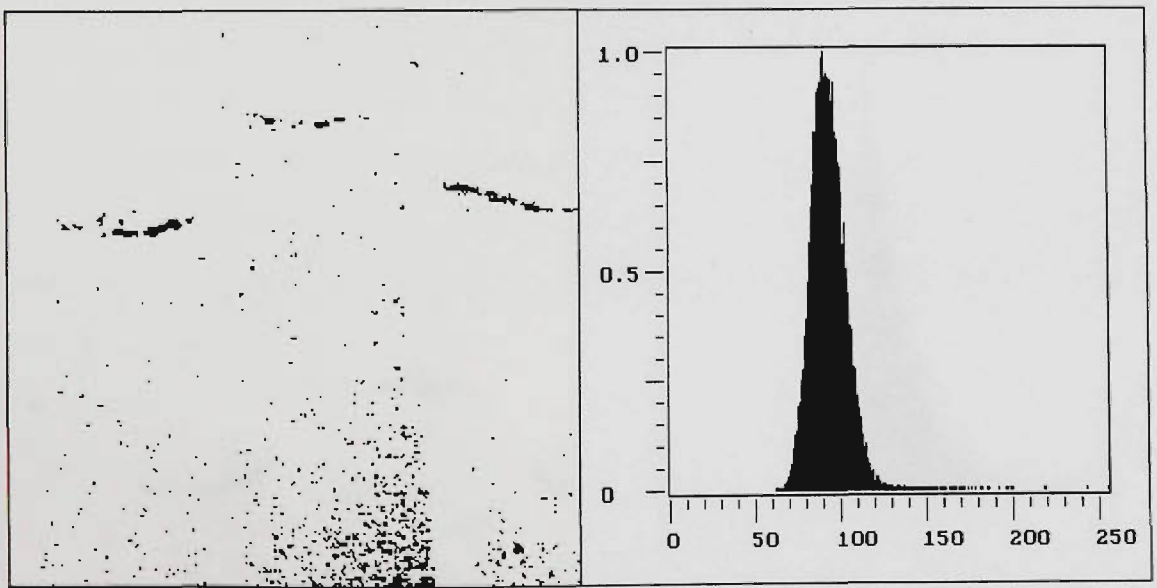
The normalised intensity histogram for the tile sample of Fig. 4.3(f) is similar to the wool histograms, with a single peak. However, even though the histogram of Fig. 4.4(f) has a discernible valley, it remains difficult to predict the lower limit intensity of the defect patterns.



(a) Poorly illuminated tile surface. (b) Segmented image ($\eta = 400$, $\nu = 150$).

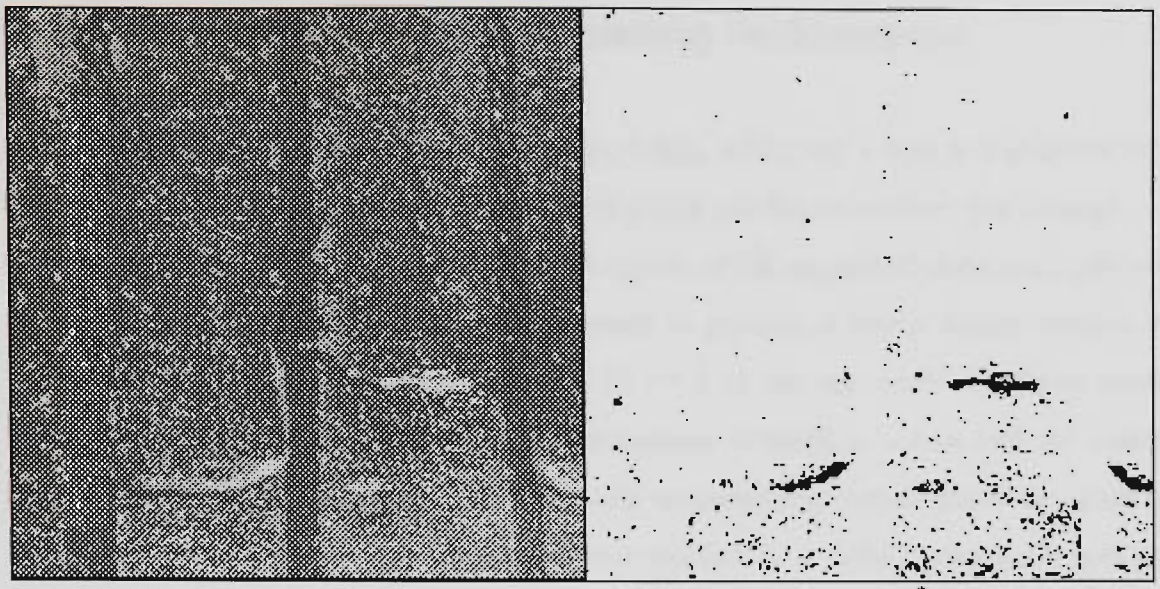


(c) Blob analysis of (b) (area > 2 pixels). (d) Three linear clusters detected.

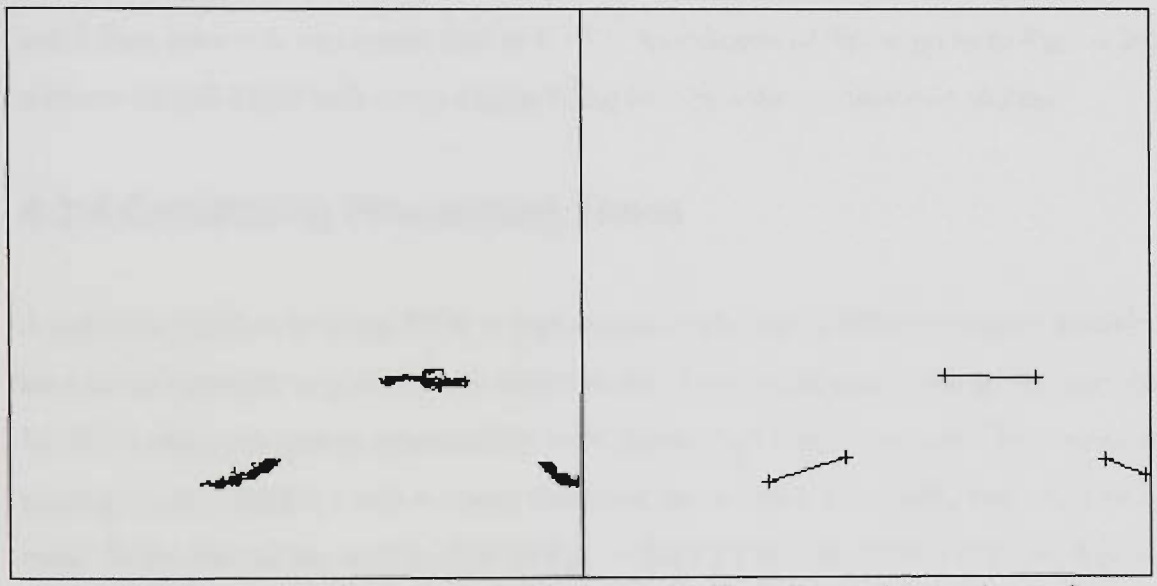


(e) FCM segmented image ($c = 12$, $\nu = 119$). (f) Gray level normalised histogram of (a)

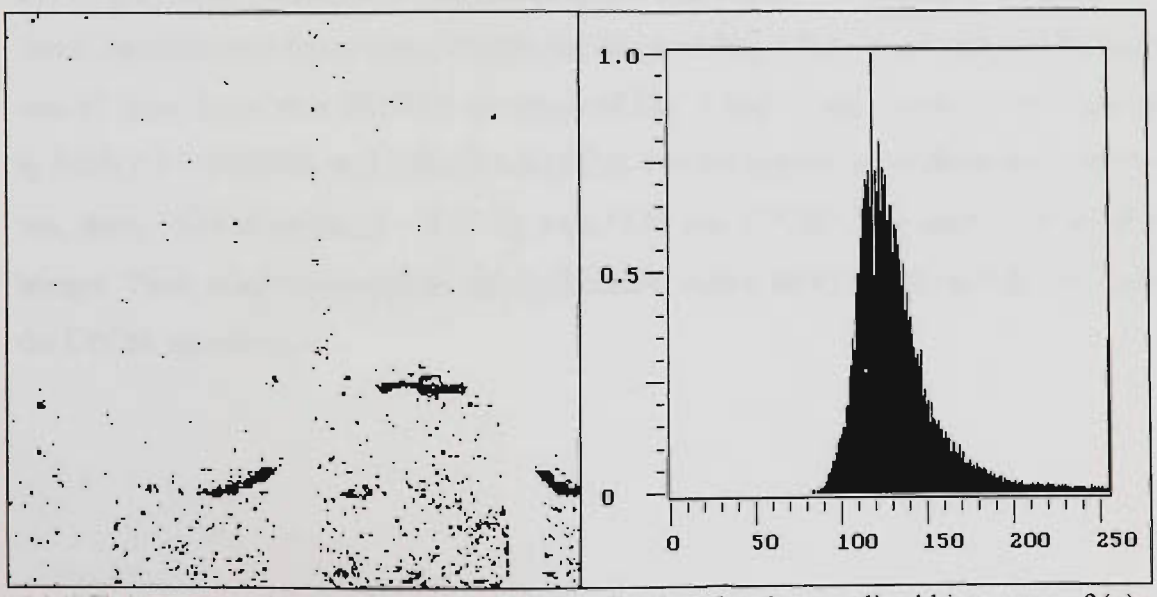
Figure 4.3 Tile at low illumination.



(a) High illumination of tile surface. (b) Segmented image ($\eta^* = 327$, $\nu = 242$).



(c) Blob analysis of (b) (area > 50 pixels). (d) Three linear clusters detected.



(e) FCM segmented image ($c = 6$, $\nu = 246$). (f) Gray level normalised histogram of (a).

Figure 4.4 Tile at high illumination.

4.3.3 Comparison of FCM Clustering Performance

Comparing the intensity images of Figs. 4.1(a), 4.2(a), 4.3(a) and 4.4(a), it is observed that FCM fails to cluster correctly when the defect points are few in number. For example, in Figs. 4.2(d) (wool) and 4.3(c) (tile), the pixel counts of the segmented blobs were 284 and 139 respectively. Moreover, this condition tends to produce a higher cluster number in FCM clustering, such as $c = 12$, compared to $c = 6$ in the successful clustering cases. When FCM clusters successfully, the cluster number required is almost half the cluster number of the difficult clustering cases, with the corresponding pattern points in excess of 1300. However, it should not be concluded from this that $c = 6$ is the optimum for both the tile and the wool defect detection problems. In fact, the FCM cluster result of Figs 4.2(a) and 4.3(a), for $c = 6$, was worse than at $c = 12$. An example of this is given in Figs. 4.2(c) with $c = 12$ and 4.2(e) with $c = 6$. Figure 4.3(a) has the same problem (not shown).

4.3.4 Comparing Processing Times

A practical problem in using FCM is that several trials with a different cluster number c have to be attempted to pick the best cluster result. More significantly, the processing time for FCM increases almost exponentially with increasing cluster numbers. The processing times given in Tables 4.1 and 4.2 were measured on an i486DX/33-MHz Intel microprocessor. In the case of the wool images of Fig. 4.1(a) CPCM was 170 times faster than the FCM, and for Fig. 4.2(a), 225 times faster than FCM. For the tile images, the best CPCM speed was 255 times faster than FCM for the image of Fig. 4.3(a). The worst CPCM speed was 41 times faster than FCM for the image of Fig. 4.4(a). These results are summarised in Table 4.1 for CPCM, and Table 4.2 for FCM. For the purpose of performance comparison, thirty FCM iterations ($\varepsilon < 0.05$ for both FCM and CPCM) were used in all the four images. These results demonstrate the significantly higher data processing efficiency from the CPCM algorithm.

Image	η	Number of pixels	v	Intensity low,high	d_η	Processing time (sec)
Fig. 4.1(a)	400	58501	81	39,123	42.14	10
		3828	140	124,182		
		2878	29	0,38		
		53	188	183,230		
	589	63627	81	19,143	62.06	8
		1315	158	144,220		
		318	13	0,18		
Fig. 4.2(a)	400	63285	124	82,166	42.14	9
		1642	71	29,81		
		353	188	167,230		
	439	63849	124	78,170	46.25	9
		1147	68	21,77		
		284	193	171,239		
Fig. 4.3(a)	400	65131	93	51,135	42.14	8
		139	150	136,193		
		10	224	194,255		
	550	65218	93	35,151	57.95	8
		58	169	152,227		
Fig. 4.4(a)	400	60605	127	85,169	42.14	9
		3644	190	170,232		
		1028	252	233,255		
	327	58238	126	92,160	34.45	13
		5391	177	161,211		
		1524	242	212,255		
		127	90	55,91		

Table 4.1 Summary of cluster statistics from CPCM algorithm. c is the number of clusters, v is the cluster prototype and d_η is the cluster radius, centred at prototype v. Note: The intensity range is represented by the lower and upper intensity limits, and do not overlap even though the prototype v is unequally spread. The order of the prototype v in the table corresponds to the order obtained from the algorithm. Processing time was measured on an i486DX/33-MHz Intel microprocessor.

Image	c	v	Processing time (sec)
Fig. 4.1(a)	10	157	1416
Fig. 4.2(a)	6	149	534
	12	155	1912
Fig. 4.3(a)	12	119	1976
Fig. 4.4(a)	6	246	533

Table 4.2 Summary of cluster statistics from FCM algorithm. c is the number of clusters and v is the cluster prototype. Processing time was measured on an i486DX/33-MHz Intel microprocessor.

4.4 Conclusions

Sequential fuzzy clustering algorithms offer certain advantages for image segmentation. The two applications, relating to the detection of wool contaminants and tile surface defects, demonstrate the simplicity, clustering efficiency and effectiveness of the CPCM region segmentation algorithm. The failure of FCM to generate reasonable prototypes close to the defect pattern for the images of Figs. 4.2(a) and 4.3(a) were a surprising discovery. The problem with FCM seems to be due to the insensitivity of the algorithm to small number of cluster points. In other words, FCM works well only if the number of defect points in a cluster exceed a minimum threshold, such as 1200 points in Figs 4.2(a) and 4.3(a). The CPCM algorithm, unlike FCM, is ideally suited for this role. This is attributed to the order by which the CPCM segmentation algorithm removes the regions, begining with the major background cluster, consisting of approximately 96 % of the original image data (see Table 4.1). The remaining 4 % of the data have a higher proportion of the defect patterns and consequently are more easily clustered. Another advantage with the CPCM segmentation algorithm is the efficient extraction of a cluster region given the prototype defined by (4.4). With FCM, the process is more computation intensive, involving two separate steps. Firstly, the membership of all points must be computed from all c prototypes and secondly, a crisp partition rule such as (2.4.14) is applied to extract the specific cluster corresponding to the prototype. The ability to efficiently generate a cluster from the prototype allows the use of hybrid schemes such as a neural network and a cluster generating algorithm (instead of a clustering algorithm) to improve clustering performance. This technique is reviewed in more detail in Chapter 8.

The particular implementation of the CPCM clustering method can also be varied to explore the possibility of more efficient clustering performance. For example, a good candidate algorithm for consideration is the variable η reviewed in Section 3.4.1. The EPCM and PFCM possibilistic algorithms could also be considered for this purpose. These are some of the numerous unexplored possibilities offered by the fuzzy clustering approach for region segmentation.

Chapter 5

Detection of Linear Clusters

To simplify the analysis, the clustering method is restricted to linear clusters that are detected as points in 2D space. The clustering method incorporates several procedures for line detection defined by the line prototype and line gradient in Section 5.2 and the line characteristics tests in Section 5.3. The line tests ensure that valid features present in the lines such as notches or slots are recognised by the algorithm. The experimental results of five edge-segmented real objects are discussed in Section 5.5. The clustering method can be extended for applications involving clusters of higher dimensions.

5.1 Introduction

Characterising the boundary from outlines is an effective method to obtain useful information from digital images of real world objects. This method simplifies the image processing task in terms of computation effort and memory requirements. A variety of line detection schemes exist in the literature [Davies, 1990; Duda and Hart, 1973; Gonzalez and Woods, 1992]. Historically the Hough transform (HT), invented by Hough in 1962, is considered to be the main method for detecting straight lines. Currently, other methods are gaining popularity. One such method is fuzzy clustering which offers higher computation efficiency with lower memory requirements.

The fuzzy c-varieties clustering algorithms [Bezdek et al., 1981a, 1981b] extended the detection capabilities of FCM to linear and planar clusters. Dave [1989] improved the linear cluster detection of non-ordered data set with the Adaptive Fuzzy Clustering algorithm (AFC). The AFC used a modified distance metric involving a mixing coefficient.

Dave showed that the AFC compared favourably to the HT, uses less memory and produces better results in the case of polygonal approximation of curves. However, the original AFC was sensitive to noise and entails intensive computation. Consequently, Dave proposed a progressive version of the AFC (this seems to be the first documented reference to a progressive fuzzy clustering algorithm) that removes good linear clusters during the clustering process. Possibilistic clustering was introduced in [Krishnapuram and Keller, 1993a] to achieve more robust and accurate clustering in the presence of noise. A possibilistic algorithm reformulates the probabilistic membership of FCM by basing membership values solely as a function of the distance of a point from its nearest prototype or centre. In 1995, the possibilistic algorithm was extended to detect hyperquadric shell structures [Krishnapuram et al., 1995].

5.2 Two Stage Clustering

CPCM is a suitable framework for implementing the fuzzy clustering methods to detect linear clusters. In the first clustering stage, CPCM obtains an approximate prototype with a round shape cluster structure. At the second clustering stage, a linear cluster prototype is detected if the points satisfy the line definition criteria.

5.2.1 First Stage Clustering

The membership equation for points in $\Re^2$ is given by

$$u_k = \frac{1}{\exp\left[\dfrac{(x_k - v_1)^2 + (y_k - v_2)^2}{\eta^2}\right]} \tag{5.1}$$

where the coordinates of the kth point $\mathbf{x}_k$ is (x_k,y_k) and the prototype coordinates $\mathbf{v}$ is (v_1,v_2). The root mean square of the Euclidean distances of points from the prototype is

$$\eta = \sqrt{\frac{s}{N_c}\sum_{k=1}^{N_c}\left[(x_k - v_1)^2 + (y_k - v_2)^2\right]} \tag{5.2}$$

where N_c is the total number of points in the current data list and s is a constant scale factor. The prototype component equations are given by

$$v_1 = \frac{\sum_{k=1}^{N_c} u_k^2 x_k}{\sum_{k=1}^{N_c} u_k^2} \tag{5.3a}$$

$$v_2 = \frac{\sum_{k=1}^{N_c} u_k^2 y_k}{\sum_{k=1}^{N_c} u_k^2} \tag{5.3b}$$

where $\mathbf{v} = [v_1, v_2]^T$.

First stage clustering algorithm.

Fix s, α_t, q and N_{min}, and assign data centroid to $\mathbf{v}$.

Repeat

Calculate u_k from (5.1) and η from (5.2).

Calculate $\mathbf{v}$ from (5.3a) and (5.3b).

Find nearest neighbour of prototype $\mathbf{v}$ and assign to $\mathbf{v}$.

Until $\|\mathbf{v}_t - \mathbf{v}_{t-1}\| < \delta$ or $t > 50$.

Note: δ is a small number to determine the stopping point and t is an iteration index.

5.2.2 Second Stage Clustering

The second stage coerces the prototype from the first stage towards a linear cluster boundary, with an additional test for a line gradient. The line gradient is defined by

$$m = \begin{cases} \dfrac{\sum_{k=1}^{N_c} \dfrac{(y_k - v_2)}{(x_k - v_1)} F_k}{\sum_{k=1}^{N_c} F_k} & \text{if} \quad F_k > 0 \\ 0 \quad \text{otherwise} \end{cases} \tag{5.4}$$

$$F_k = \begin{cases} 1 & \text{if} \quad u_k > \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

where α_t is the alphacut threshold on membership. The line gradient is used in a membership equation defined by

$$u_k = \frac{1}{\exp\left[\dfrac{|y_k - v_2 - m(x_k - v_1)|^q}{\eta^2}\right]} \tag{5.5}$$

where η is defined by (5.2) and q is a real valued exponent. The prototype equations are identical to (5.3a) and (5.3b).

Second stage clustering algorithm.

Use initial **v** from the first stage clustering.

Repeat

Calculate m from (5.4).

Calculate u_k from (5.5) and η from (5.2).

Calculate **v** from (5.3a) and (5.3b).

Find nearest neighbour of **v** and assign to **v**.

Until $\|\mathbf{v}_t - \mathbf{v}_{t-1}\| < \delta$ or $t > 50$.

Note: δ is a small number to determine the stopping point and t is an iteration index.

The point-slope form of the line equation (5.4) requires special treatment as the slope approaches infinity. One way of dealing with this condition is to accumulate the points with very small differences in the horizontal axis and later assign these points with an arbitrary large gradient. Consequently, it is necessary to check for infinite gradient condition prior to calculating the line gradient. The CPCM framework supervises the clustering process to ensure that only a valid cluster is developed and manages the data for efficient processing.

5.3 Cluster Merging and Corner Detection

Discontinuity in a line may arise in a number of ways that occurs naturally in features of the object or from the edge segmentation process. Consequently, this information needs to be considered by the clustering algorithm so that real line breaks are correctly recognised and spurious ones are ignored. A continuous line is defined according to the rule: If the pixel gap $g \leq pgap$ (specification) then the line is continuous. The pixel gap is defined as the distance between two ordered adjacent pixels.

One algorithm for checking a continuous line is given by the following procedure in pseudocode, assuming the candidates of line points in the horizontal axis have been first sorted in an ascending order.

For (i = 1 to N_c) **do**

Begin

If ($\mathbf{x}_{i+1} - \mathbf{x}_i > pgap$) **Then**

If ($xcount < N_{min}$) **Then** Start a new line and discard previous accumulated points.

Else Continuous line found and exit procedure.

Else Accumulate $\mathbf{x}_i$ as a line point and increment *xcount*.

End.

The symbols *xcount*, N_c and N_{min} represent the count of $\mathbf{x}_i$ as a line point, the current data count, and the minimum number of cluster points respectively. The loop control variable is denoted by the index *i*. The test for *xcount* is made to ensure that only a valid cluster (points > N_{min}) will constitute a continuous line.

To assess suitability for cluster merging, we apply three tests in the following order: (i) slope ratio test, (ii) line separation test and (iii) line ends gap test. The slope ratio test evaluates the sign of the ratio of line gradients. If the ratio is positive, then the lines are candidates for merging. The line separation test involves two line equations as shown in Fig. 5.1, where the points (x_1,y_1) and (x_2,y_2) are the line prototypes and m_1, m_2 are their corresponding gradients.

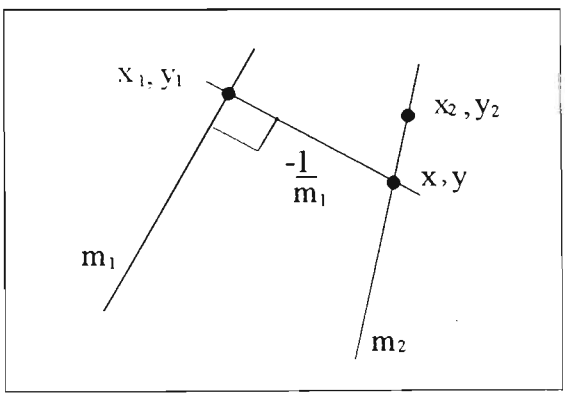


Figure 5.1 Line separation distance definition.

$$x = \frac{y_1 - y_2 + \frac{x_1}{m_1} + m_2 x_2}{m_2 + \frac{1}{m_1}} \tag{5.6}$$

$$y = y_2 + m_2(x - x_2) \quad \text{if } m_1 > m_2 \tag{5.7}$$

$$y = y_1 - \frac{1}{m_1}(x - x_1) \quad \text{if } m_1 \le m_2 \tag{5.8}$$

The line separation distance is defined as

$$g = \sqrt{(x - x_1)^2 + (y - y_1)^2} \tag{5.9}$$

For g calculated from (5.9), if $g \leq lgap$ (specification), then the lines are considered to be candidates for merging. The line ends gap is defined as the gap between the two nearest ends of the two lines. If this gap, $g \leq egap$ (specification), then the two lines are candidates for merging. In consequence of merging the clusters (may involve more than two clusters), the prototype position and the gradient for the new line have to be calculated. At this point, a least squared fit of the line or other robust line regression methods given in [Lawrence and Arthur, 1990] can be applied to improve line detection in the presence of noise. Our algorithm implements a standard least squared error fit of the cluster points.

Once a continuous line has been established, the detection of corners is relatively straight forward. For corner detection, we apply the rule that the closest ends of two lines form a corner. In case the ends of the lines are intended to be open ended, we can apply a gap test to ensure that corners are only found for line ends in close proximity. This corner prescription assumes all corners are defined likewise, a reasonable simplification used in this application. Fig. 5.2 shows the line relationships for corner calculation. Points (x_1, y_1) and (x_2, y_2) are the line prototypes and m_1, m_2 are their corresponding gradients.

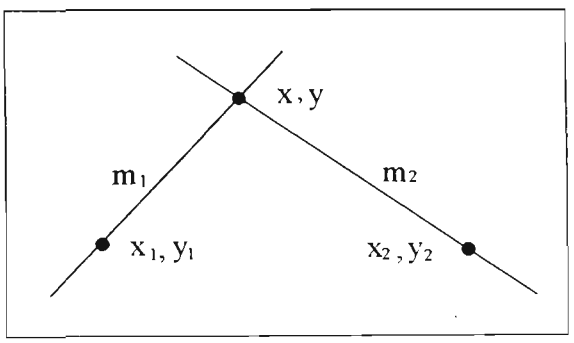


Figure 5.2 Line corner definition.

$$x = \frac{y_1 - y_2 - m_1 x_1 + m_2 x_2}{m_2 - m_1} \tag{5.10}$$

$$y = y_2 + m_2(x - x_2) \quad \text{if } m_1 > m_2 \tag{5.11}$$

$$y = y_1 + m_1(x - x_1) \quad \text{if } m_1 \leq m_2 \tag{5.12}$$

The corner gap is the minimum distance defined by

$$g = \min\left(\sqrt{(x - x_1)^2 + (y - y_1)^2} \text{ or } \sqrt{(x - x_2)^2 + (y - y_2)^2}\right) \tag{5.13}$$

If $g \leq cgap$ (specification), then the corresponding lines have a candidate corner.

5.4 Pseudo Code

The pseudo code for linear boundary detection and the eight parameters are given below.

Pseudo code to detect linear clusters.

Initialise α_t, N_{min}, q, s, $pgap$, $lgap$, $egap$, and $cgap$.

Compute initial **v** from mean data centroid and η from (5.2).

Repeat

First stage clustering loop (from Section 5.2.1).

Second stage clustering loop (from Section 5.2.2).

If $(0 \leq N_\alpha < N_{min})$ **Then** Remove N_α from data and update N_c.

If $(N_\alpha >= N_{min})$ **Then** Save cluster parameters (line ends, slope and prototype),

remove cluster points of prototype **v** from data and update N_c.

Until $(N_c < N_{min})$

The symbols N_{min}, N_c and N_α represent the minimum points in a cluster, the current points count of data list and the number of points in the current cluster, respectively. For the four objects used in the experiments. satisfactory clustering performance was obtained by presetting eight of the parameters to the following constant values: $\alpha_t = 0.95$, $N_{min} = q = lgap = 5$, $pgap = egap = cgap = 10$ and, $s = 0.5$. Parameter values for object number *5* required different settings. All constants, except for s, have units in pixel.

The different types of cluster tests used in the linear cluster detection algorithm are summarised below:

1. *Cluster validity test*: If cluster points, $N_\alpha \geq N_{min}$ then a valid cluster is found.
2. *Line continuity test*: If pixel gap, $g \leq pgap$ then the points constitute a continuous line.
3. *Line merge criteria*:

   The lines are merged if all three criteria, in the order shown, are satisfied.

   (i) If ratio of line slopes (maximum = 1), $mratio > 0$ and

   (ii) If the line separation distance, $g \leq lgap$ and

   (iii) If ends of lines gap, $g \leq egap$, then the lines are merged.
4. *Corner test*: If the two nearest ends of the lines, $g \leq cgap$, then a corner exist.

5.5 Characteristics of Cluster Parameters

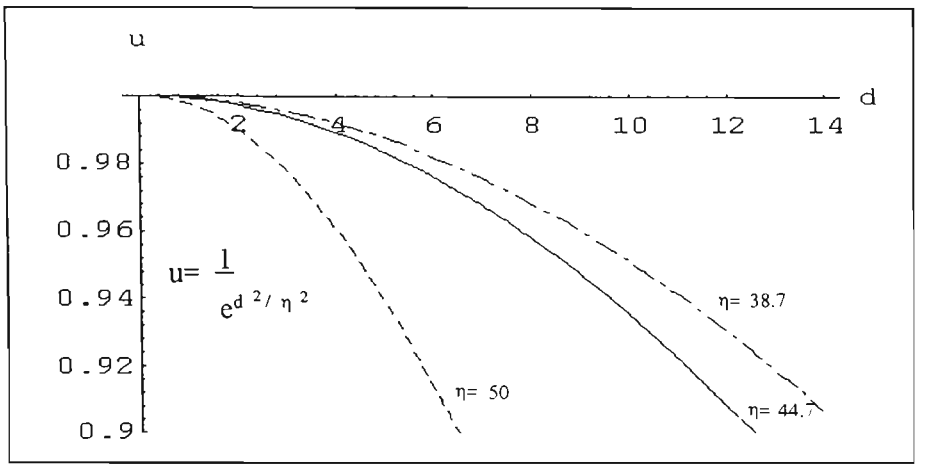


Figure 5.3 First stage membership functions from equation (5.1).

At a fixed membership u_k and the distance exponent q, the cluster radius d decreases with increasing values of η as indicated in Fig. 5.3. This means that smaller η values are easier to cluster because they have more points above the alphacut α_t. However, the cluster results are not likely to be as accurate as those at larger η values. Since the first stage clustering is only expected to locate an approximate cluster prototype, accuracy of cluster prototype location is not essential at this stage. In general, the upper and lower bounds of η depend on the data.

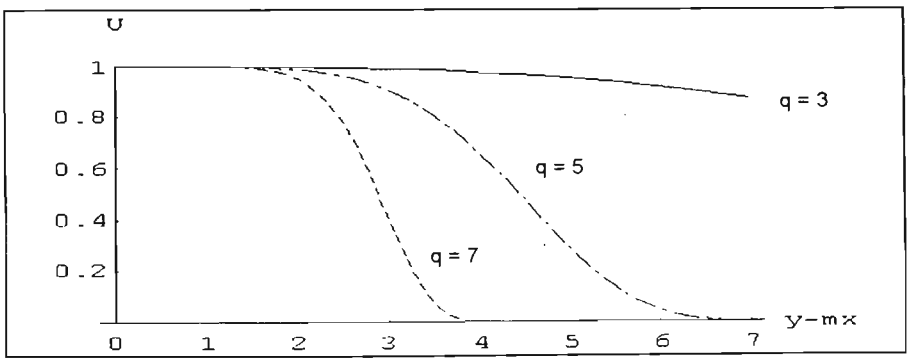


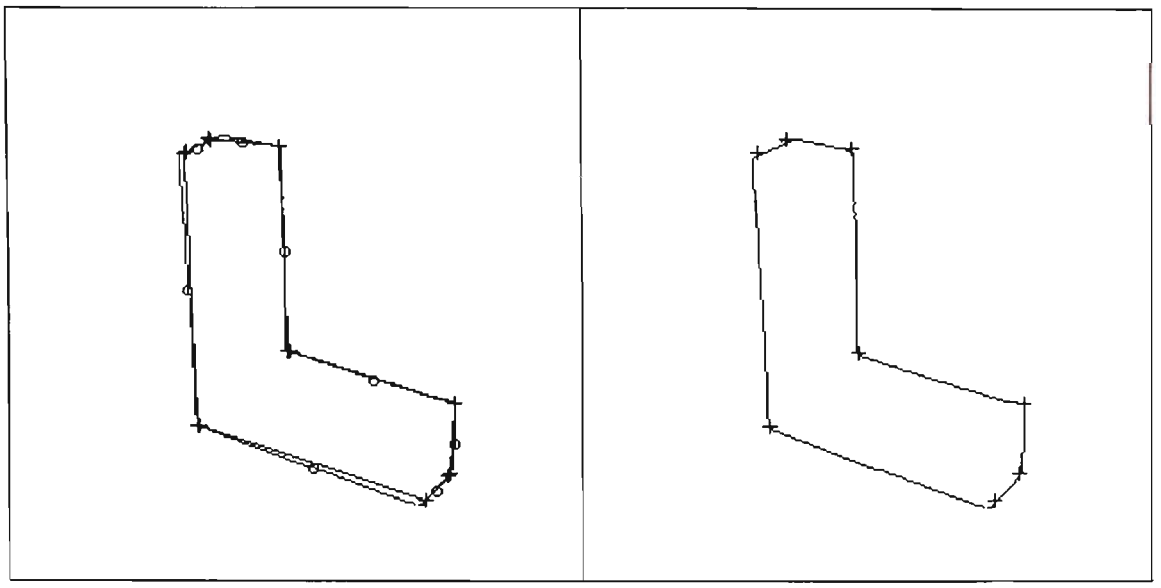
Figure 5.4 Second stage membership functions from equation (5.5), at fixed η.

The second stage clustering refines the approximate prototype **v** from the first stage by applying an alphacut α_t on the membership and looks for a line gradient from (5.4). Figure 5.4 shows that the characteristics of the membership is shaped by the values of q. Like the first stage clustering, smaller q values are easier to cluster because they yield more cluster solutions. Larger q values generally produce less cluster solutions but more accurate cluster parameters such as the line gradient and the location of the prototype. The

inclusion of procedures for compatible cluster merging in the algorithm design supports high q values for more accurate clustering. Normally, high q clustering results in fragmented clusters of small line segments. To remedy this, compatible clusters are merged according to the three criteria given in Section 5.4, producing a more compact cluster structure. Examples of cluster merging are demonstrated in Section 5.6.

5.6 Experimental Results

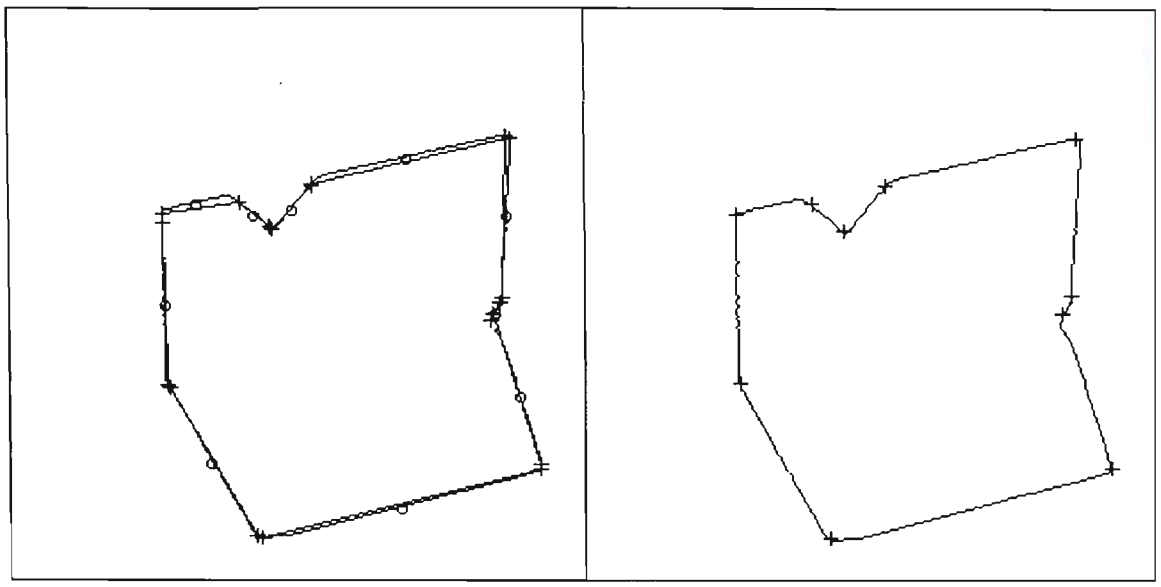
The linear cluster detection algorithm requires input data format as outlines of an object. This is obtained from any of the conventional methods of edge segmentation discussed in Section 7.3. The algorithm was evaluated on five different object outlines, numbered *1* to *5*. All objects were acquired in 2D plan view and in 256 × 256 resolution, except for object *number 5*, which has a resolution of 300 × 260. Object number *1* (Fig. 5.5) tests line detection performance. It is an *L*-shape block with long and short line lengths that are quite well defined. Object *2* (Fig. 5.6) is almost square shape, also with well defined long and short lines but differentiated by a medium size notch on the top side and a smaller notch on the right side. The *v*-notches test the algorithm for correct recognition of lines on either side of the notch. Object *3* (Fig. 5.7) is a poorly defined pentagon in a noisy background to assess the impact of noise on cluster recognition and detection accuracy. Object *4* (Fig. 5.8) is a silhouette of an electric motor armature housing, containing edge noises and fragmented outlines associated with typical problems of low level edge segmentation. This image is provided to test the algorithm's performance under real world conditions. Object *5* (Fig. 5.9) contains multiple intersections of complicated outlines with a mixture of single and double lines, to test corner detection accuracy.



(a) Eight clusters detected. (b) Eight corners detected.

Figure 5.5 Object number *1* (475 points) at fixed cluster parameters of Table 5.2.

All eight clusters and corners of Figure 5.5 were detected correctly. The cluster prototype is computed as the mean of the cluster points. It is denoted by a small circle, located at or near the mid point of the cluster line. The lines of the clusters are superimposed on the object's outline to assist comparison. The end of each cluster line is indicated by a small cross. A common problem with a line detection algorithm is the creation of false lines criss-crossing the image in a random manner. The absence of this problem in Fig. 5.5 is attributed to the effectiveness of the line continuity check, and also partly to the success of the compatible cluster merging algorithm. Small gradient errors are noticeable in Fig. 5.5(a), resulting from the general choice of fixed cluster parameters of Table 5.2. Increasing q or using higher α_t is expected to improve accuracy of the cluster result, for the reasons given in Section 5.5.

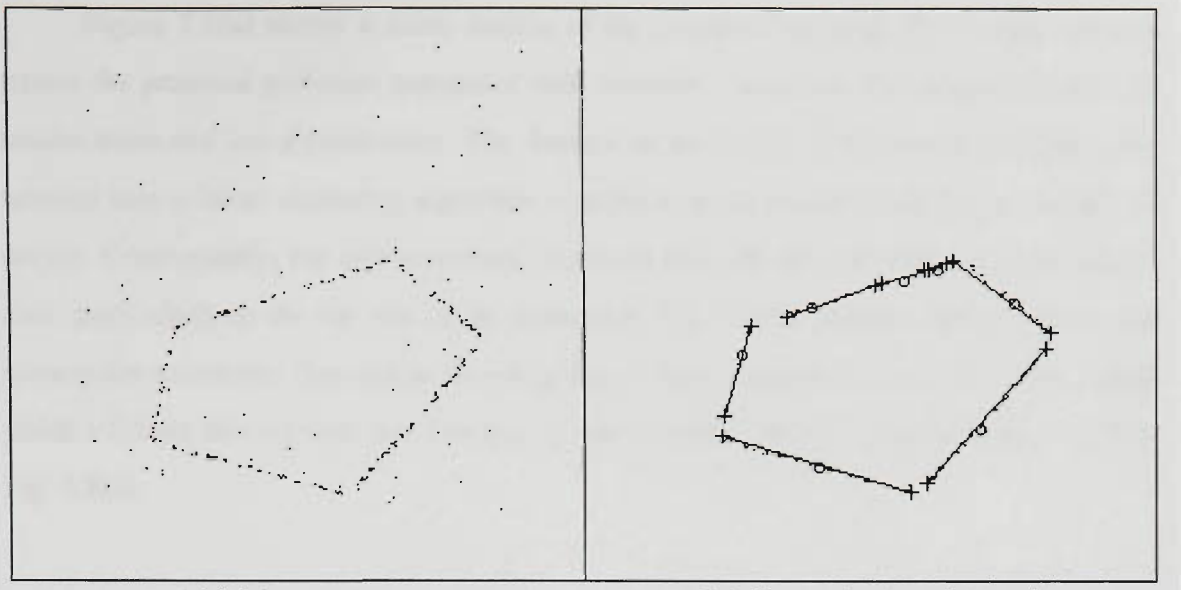


(a) Ten clusters detected. (b) Ten corners detected.

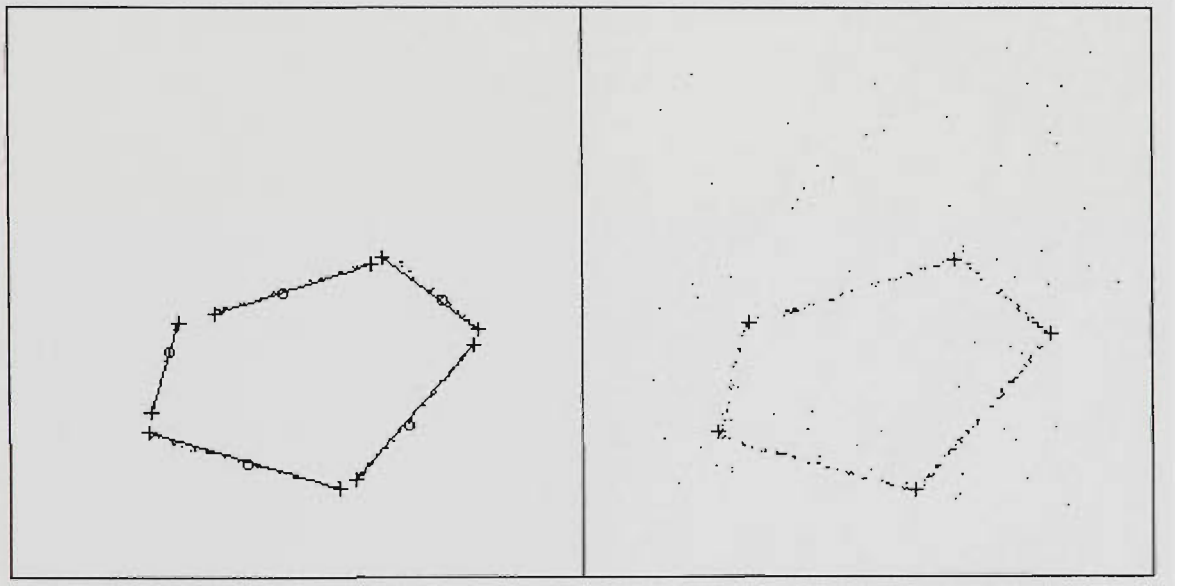
Figure 5.6 Object number *2* (576 points) at fixed cluster parameters of Table 5.2.

Figure 5.6(a) shows two line notches correctly detected. The line gaps from both notches are 35 and 17 pixels, respectively. The correct identification of corners is shown in Fig. 5.6(b). Note that the lower corner of the smaller notch only appear misplaced because of the way the adjacent lines intersect. As in Fig. 5.5, the algorithm correctly detects vertical lines.

Figure 5.7(a) is a noisy image of a pentagon. The absence of distinct lines in the image would present considerable problems for most boundary tracking algorithms in the search for edge and corner features. The CPCM based linear fuzzy clustering algorithm correctly identifies the five major clusters in Fig. 5.7(c) and the five corners in Fig. 5.7(d). Figure 5.7(b) shows the initial clusters found from the fixed cluster parameters. The top edge shows three suitable candidate clusters for merging. The result of cluster merging is shown in Fig. 5.7(c). Note the prototype position after cluster merging. All five corners of the fragmented pentagon are correctly identified from the intersection of adjacent lines. Given the noisy condition of the original image, the line gradients of Fig. 5.7(c) are acceptable. Note that the clean image of Fig. 5.7(c), compared to Fig. 5.7(a), is a feature of CPCM which automatically removes noise points to sustain cluster development. This is a reason why a CPCM based fuzzy clustering algorithm is more computation efficient, compared to other non-sequential fuzzy clustering algorithms. Table 5.1 summarises the line statistics of the pentagon objet, relating to the number of points in each cluster (line), prototype positions, slope and corners.



(a) Noisy pentagon. (b) Seven clusters detected.



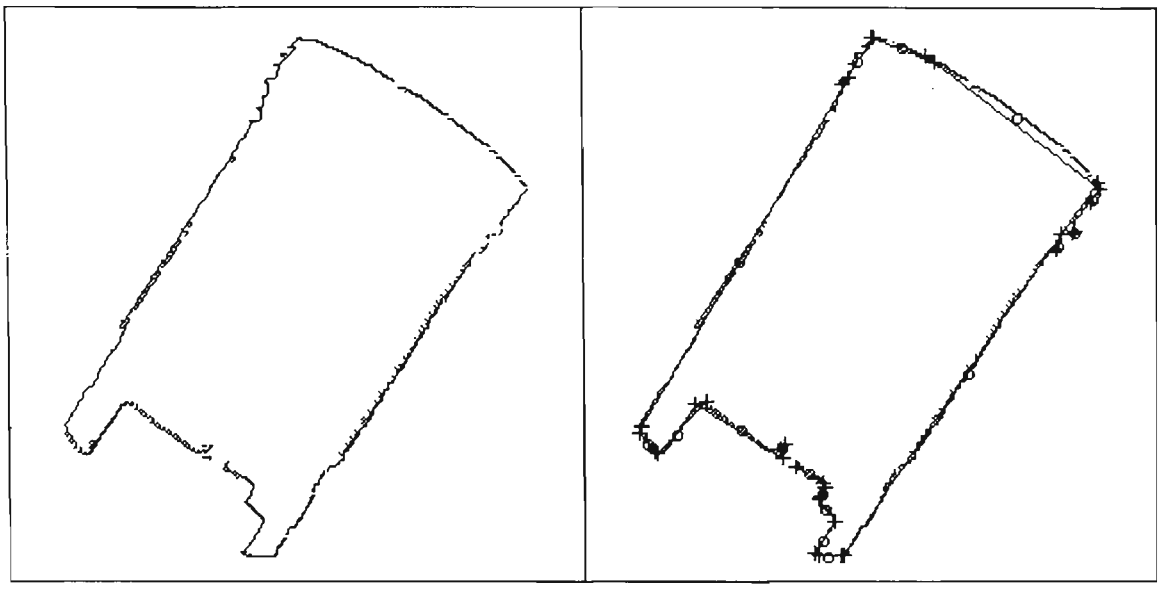
(c) Five clusters detected after merging. (d) Five corners detected.

Figure 5.7 Object number *3* (252 points) at fixed cluster parameters of Table 5.2. Note: the cluster centre is denoted by a small circle and the corners and ends by small crosses.

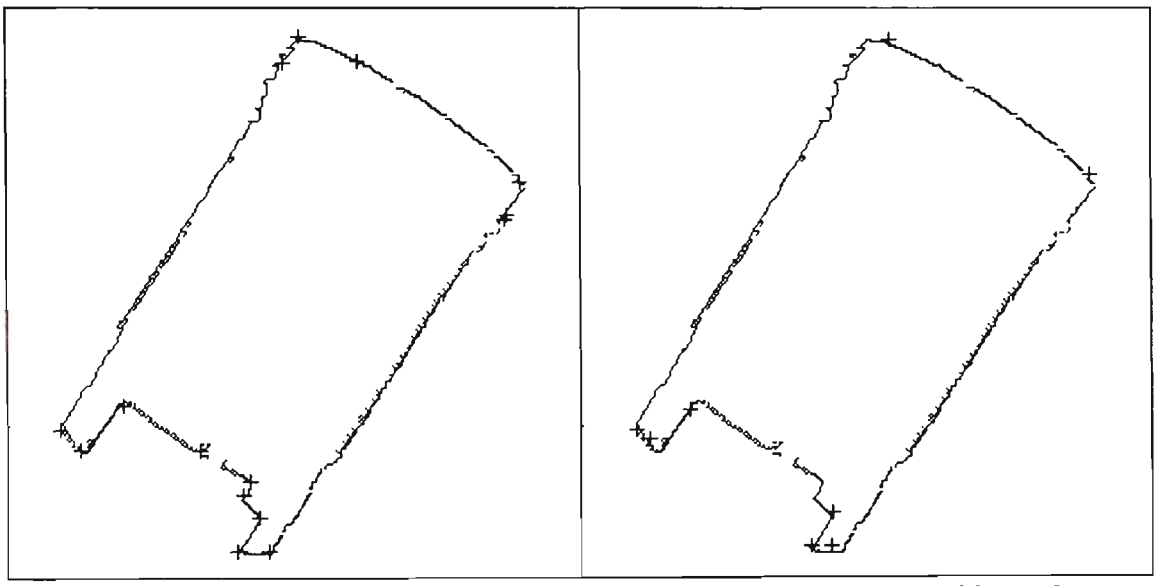
Cluster number	Number of points	Centre (pixel units)	Slope
1	51	176, 69	1.12
2	30	120, 128	0.224
4	31	191, 125	-0.813
5	42	104, 52	-0.277
7	8	70, 102	3.58

Table 5.1 Summary of the 5 line statistics of Fig. 5.7(c). The corner dimensions of Fig. 5.7(d) are: (209, 109), (148, 39), (166, 142), (74, 114) and (60, 65). The origin of the co-ordinates is at the lower left corner of the image.

Figure 5.8(a) shows a noisy outline of an armature housing. This image demonstrates the practical problems associated with boundary detection. The edges indicate extensive noise and line discontinuity. The clusters shown in Fig. 5.8(b) are acceptable, considering that a linear clustering algorithm is applied on an object which has a number of curves. Consequently, the approximation of curved sides by straight lines are quite noticeable, particularly at the top side of the housing in Fig. 5.8(b). Despite this condition, the corners are accurately detected as shown in Fig. 5.8(c). A smaller q ($q = 3$) in Fig. 5.8(d) yields a fuzzier line segment and thus less accurate corner detection compared to $q = 6.5$ in Fig. 5.8(c).



(a) Outline of noisy armature housing. (b) Clusters identified with $q = 6.5$.



(c) Corners detected with $q = 6.5$ (d) Corners detected with $q = 3$.

Figure 5.8 Object number *4* (1196 points) at fixed cluster parameters of Table 5.2, except for the indicated q parameter.

Figure 5.9(a) shows a more complicated image for corner detection. This condition requires a different set of cluster parameters from those used in Figs. 5.5 to 5.8. Figure 5.9(b) shows that all the main cluster features are correctly identified, including the two corners of the outer cluster boundary. However, the four interior corners of Fig. 5.9(b) formed by the intersections of double lines exhibit noticeable errors. This is apparent from the two non-parallel lines linking the corners in Fig. 5.9(c). This problem illustrates that real world images have certain features that can be difficult to generalise with a clustering algorithm. In this particular case, the problem is due to the method which located the line ends solely from a line continuity test. For line ends defined by the two points (x_1,y_1) and

(x_2,y_2), one possible solution is to use the merged line gradient m and prototype $\mathbf{v}$ to obtain a better estimate of the line ends along the y axis with

$$y_1 = m(x_1 - v_1) + v_2 \tag{5.14}$$

$$y_2 = m(x_2 - v_1) + v_2 \tag{5.15}$$

where x_1 and x_2 are obtained from the line continuity test procedure. This modification produces an improved corner detection shown in Fig. 5.9(d). Unfortunately in this case, there is substantial loss of line details, particularly at the right side interior lines, because of unwanted side effects from the modification. Depending on the particular application specifications, this condition may be satisfactory. For example, if corner detection is the main objective. Suppose the application specification requires both good line and corner detection. In this case, a further analysis of the procedure indicates that the above modification provides only a partial solution. A closer scrutiny reveals that Eqs. (5.14) and (5.15) do not give accurate prediction of line end-points for the case where the gradient is large, because of insufficient resolution in the x axis direction. Thus a small error in the x axis is accentuated along the y axis. Clearly good end-point estimates for *both* low and high gradients will rectify the problem. This provides the solution to the following revised procedure:

If line slope, $m \leq 1$ then estimate y_1 and y_2 from x_1, x_2, $\mathbf{v}$ and m with

$$y_1 = m(x_1 - v_1) + v_2 \tag{5.16}$$

$$y_2 = m(x_2 - v_1) + v_2 \tag{5.17}$$

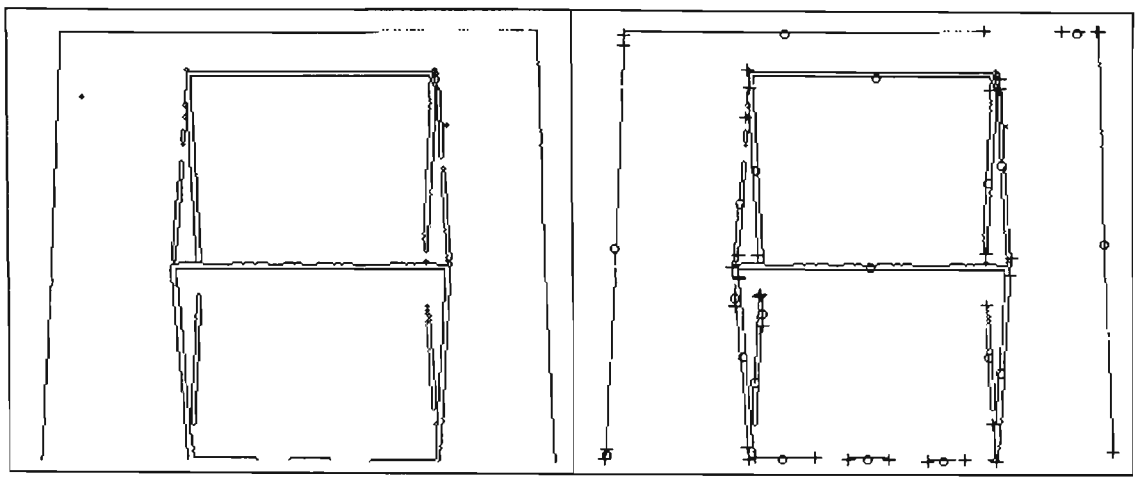
else estimate x_1 and x_2 from y_1, y_2, $\mathbf{v}$ and m with

$$x_1 = v_1 + \frac{y_1 - v_2}{m} \tag{5.18}$$

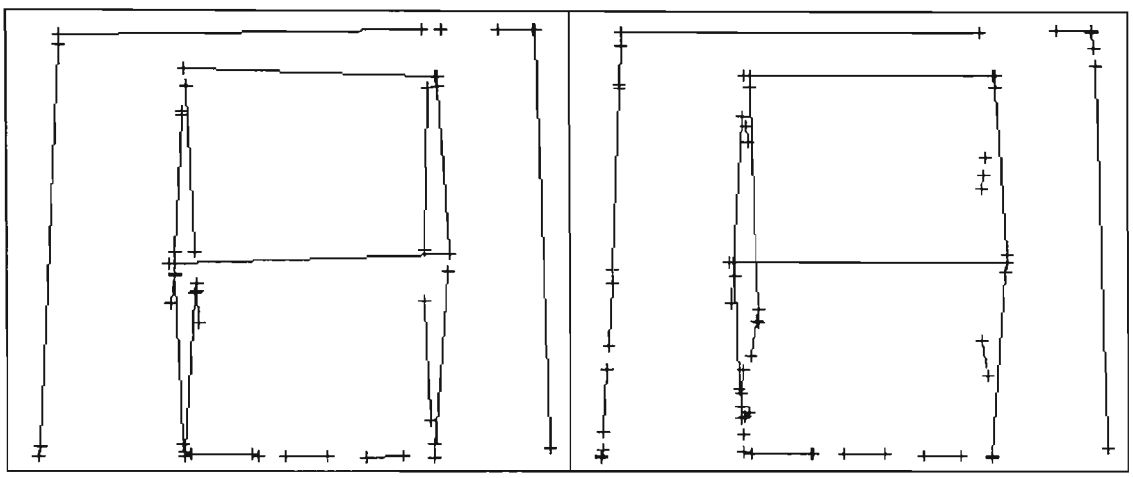
$$x_2 = v_1 + \frac{y_2 - v_2}{m} \tag{5.19}$$

The result of the revised modification is shown in Figs. 5.9(e) and 5.9(f). The improved line and corner detection of Fig. 5.9(f) compared to Fig. 5.9(d) confirms the substantial validity of the more extensive analysis.

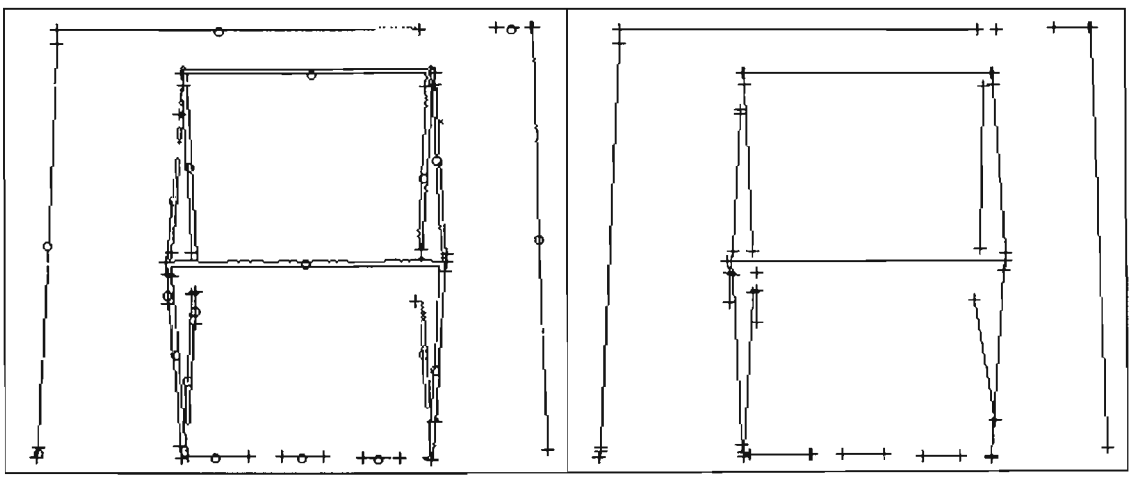
This example illustrates some of the practical problems of implementing an algorithm for application. More importantly, it indicates that the algorithm design must be sufficiently flexible to address the needs of the particular application. It may be observed that the specification of a small $pgap = 7$ correctly detects the line breaks at the top and bottom edges of Figs. 5.9(c) and 5.9(f).



(a) Object outline. (b) Detected clusters.



(c) Detected corners. (d) Improved corner detection.



(e) Improved cluster detection. (f) Improved corner detection and lines.

Figure 5.9 Object *number 5* (2568 points) with the following cluster parameters: $N_{min} = 7$, $\alpha_t = 0.9$, $q = 4$, $\eta = 89.44$, *pgap* = 7, *mratio* > 0, *egap* = 7, *lgap* = 5 and *cgap* = 10. Note that η^2 is fixed at 8,000.

Object number	Number of points	η	α_t	Processing time (sec)
1 (Fig. 5.5a)	475	45.3	0.95	1.5
2 (Fig. 5.6a)	576	60.3	0.95	3.5
3 (Fig. 5.7a)	252	48.9	0.95	1.5
4 (Fig. 5.8a)	1196	61.3	0.90	10.9

Table 5.2 Summary of processing times. The cluster parameters of all four objects were fixed to the following values: α_t =0.95, $N_{min} = q = lgap = 5$, $pgap = egap = cgap = 10$ and, $s = 0.5$. All constants, except for s, have unit of pixel. Note that η is determined from Eq. (5.2). Processing time was measured on an i486DX/33-MHz Intel microprocessor.

Table 5.2 summarises the processing time to cluster each of the four objects on an i486DX/33-MHz Intel microprocessor. It indicates that the execution time for objects of about 500 points averages 2 seconds. However, doubling the number of points increases the time by a factor of 5. For small sizes up to about 500 points, the clustering speed is reasonably fast.

5.7 Conclusions

A new method for detecting linear boundary and corners has been presented. It is based on a fuzzy clustering approach involving two clustering stages within the CPCM. The first stage centres the prototype at a round shape dense cluster to enable detection of a linear cluster by the second stage. Five objects have been used to test the algorithm's clustering performance in noisy environments. The results demonstrate accurate clustering of lines and the detection of corners. Processing time for clustering was reasonably good. The accommodation of specific line termination conditions demonstrates the flexibility of the CPCM framework. By modifying the distance measure d_{ik}, it is possible to detect other cluster structures such as hollow circles or shells. These examples indicate that CPCM can be extended in ways that will support more sophisticated clustering. The ability to do so with relative ease, as demonstrated in the example of Fig. 5.9, is essential to address the particular needs of applications. Unlike the non-sequential fuzzy clustering algorithms, it is possible to preset the cluster parameters of the boundary detection algorithm so that cluster numbers and parameters such as the prototypes, line gradients and the positions of corners are determined automatically, for a particular type of cluster structure. This feature contributes to the algorithm's ease of use and utility.

Chapter 6

Detection of Circular Clusters

A new method for the detection and parameter estimation of 2D ring shape cluster is presented. The compositional fuzzy clustering approach is based on a supervised synthesis of independently optimised fuzzy membership, cluster radius and cluster centre. Some distinctive performance features of the algorithm are (i) automatic detection of optimum cluster numbers, (ii) generalisation of detection from constant parameters and (iii) accurate clustering in the presence of noise. Tests on several edge-segmented images of real world objects consistently produced optimal cluster size with accurate estimation of their centres and radii. The algorithm provides an alternative approach to solve analytically intractable clustering problems and can be extended for the detection of clusters of higher dimensions.

6.1 Introduction

Historically the original fuzzy ISODATA, termed the Fuzzy C-Means (FCM) by Bezdek [1973] and Dunn [1973], is considered to be the basis of most fuzzy clustering algorithms. The FCM is an unsupervised clustering algorithm that finds optimum fuzzy c partitions of the feature space by iterative minimisation of a least squared objective function, subject to the constraint that membership in all clusters sum to one. It has been successfully employed to find lines and surfaces [Bezdek et al., 1981a and 1981b] and the characterisation of variable cluster shapes in a multi-dimensional feature space [Gustafson and Kessel, 1979; Gath and Geva, 1989]. Recently, the capabilities of FCM were extended to enable detection of ring shape clusters by a modification of the objective function [Man and Gath, 1994; Krishnapuram et al., 1995]. Whilst excellent results have been reported, there are practical problems in applying these algorithms. Optimal performance requires firstly, a

careful choice of initial conditions and secondly, cluster validity is needed to establish optimum cluster size. The proposed method presents a simpler scheme for the selection of cluster parameters and circumvents cluster validity verification.

6.2 Compositional Fuzzy Clustering

In the compositional fuzzy clustering approach, the approximation of analytic optimisation of the objective function is achieved by a composition of three separate optimisations involving the fuzzy memberships, the cluster radius and the cluster prototype. Optimisation of each of the three cluster parameters is supervised by CPCM to ensure optimal cluster development. Since each cluster parameter is independently optimised, there is considerable freedom in defining the membership function and the methods for cluster radius and prototype convergence. Consequently, a simpler form of the membership function is possible and more flexible rules for cluster radius and prototype convergence may be used. These are described in the next section.

6.2.1 Update of Fuzzy Membership Function

The points $\mathbf{x}_k$ in $\Re^2$ are denoted by coordinates (x_k,y_k) and prototype $\mathbf{v}$ by the coordinates (v_1,v_2). The cluster radius r has centre coordinates at the point (r_1,r_2). Membership for the cluster is

$$u_k = \frac{1}{\exp\left[\left|\sqrt{(x_k - v_1)^2 + (y_k - v_2)^2}\right|^q - r\right/\eta]} \tag{6.1}$$

where the cluster radius is initially approximated by the two component equations

$$r_1 = \frac{\sum_{k=1}^{N_c} u_k^2 (x_k - v_1) F_k}{\sum_{k=1}^{N_c} u_k^2 F_k} \tag{6.2}$$

$$r_2 = \frac{\sum_{k=1}^{N_c} u_k^2 (y_k - v_2) F_k}{\sum_{k=1}^{N_c} u_k^2 F_k} \tag{6.3}$$

$$F_k = \begin{cases} 1 & \text{if} \quad u_k > \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

$$r = \sqrt{r_1^2 + r_2^2} \tag{6.4}$$

and after applying the Optimal Circle Fit equations of (6.14) and (6.15), the cluster radius has a more accurate expression given by (6.8) in Section 6.2.2, if $N_\alpha > N_{min}$. Recall that N_α is the number of points in the cluster, and N_{min} is the minimum number of points constituting a cluster. Symbol η is a factor to scale the metric so that q lies in a convenient range. For this application η is assumed to be 3500, unless otherwise indicated. Note that (6.2) and (6.3) are similar in form to (2.4.60) for $m = 2$ (see [Man and Gath, 1994]), except for the alphacut function F_k. The iterative form of the membership equation of (6.1) is given by

$$u_{k,t} = \frac{1}{\exp\left[\dfrac{\left|\left\|\sqrt{(x_k - v_{1,t-1})^2 + (y_k - v_{2,t-1})^2}\right|^q - r_{t-1}\right|}{\eta}\right]} \tag{6.5}$$

where the iterated variable is indicated by an additional index t. Index k denotes the kth position of a feature point in the data set.

Initial v_0 is assumed to be the data centroid and initial r_0 is zero. The form of (6.1) satisfies the two limits of the fuzzy cluster membership. At the upper limit of maximum membership, the radius from points of the cluster coincides with the prototype radius r. At the extremity of non-coincidence, the membership tends toward zero. Unlike the analytic optimisation methods, the exponent q is not required to be differentiable and solvable. Optimisation of the cluster radius and centre are discussed in the next section.

6.2.2 Update of Cluster Radius

Update of the cluster radius r is made by using the fuzzy membership (6.1) and the cluster centres **v**. Two forms of the radius equations are used. The first set is used to approximate the cluster radius. The iterated components are given by

$$r_{1,t} = \frac{\sum_{k=1}^{N_c} u_k^2 (x_k - v_{1,t-1}) F_{k,t-1}}{\sum_{k=1}^{N_c} u_{k,t-1}^2 F_{k,t-1}} \tag{6.6}$$

$$r_{2,t} = \frac{\sum_{k=1}^{N_c} u_k^2 (y_k - v_{1,t-1}) F_{k,t-1}}{\sum_{k=1}^{N_c} u_{k,t-1}^2 F_{k,t-1}} \tag{6.7}$$

$$F_{k,t-1} = \begin{cases} 1 & \text{if} \quad u_{k,t-1} > \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

The second set of equations is used when the cluster radius is defined more accurately with the Optimal Circle Fit (OCF) equations [Chaudhuri and Kundu, 1993], given in Section 6.2.3. The OCF equations (6.14) and (6.15) calculate v_1 and v_2 respectively, from which the cluster radius can be updated directly as

$$r = \sqrt{\frac{\sum_{k=1}^{N_c} \left[(x_k - v_1)^2 + (y_k - v_2)^2 \right] F_k}{\sum_{k=1}^{N_c} F_k}} \tag{6.8}$$

$$F_k = \begin{cases} 1 & \text{if} \quad u_k > \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

where the alphacut α_t is the membership threshold and N_c represents the number of points in the current data list.

6.2.3 Update of Cluster Centre

Update of the cluster centre is accomplished in two stages. The first stage obtains an approximate cluster centre from the Centre Approximating (CA) equations (derived in Appendix E) given by their component equations

$$v_{1,t} = \frac{1}{N_c} \sum_{k=1}^{N_c} \left[x_k + h_k (v_{1,t-1} - x_k) \right] \tag{6.9}$$

$$v_{2,t} = \frac{1}{N_c} \sum_{k=1}^{N_c} \left[y_k + h_k (v_{2,t-1} - y_k) \right] \tag{6.10}$$

The ratio of approximate prototype radius to previous estimated radius is

$$h_t = \frac{r_t}{r_{e,t}} \tag{6.11}$$

The denominator and numerator expressions are defined by

$$r_{e,t} = \sqrt{(x_k - v_{1,t-1})^2 + (y_k - v_{2,t-1})^2} \tag{6.12}$$

$$r_t = \sqrt{r_{1,t}^2 + r_{2,t}^2} \tag{6.13}$$

where $r_{1,t}$ and $r_{2,t}$ are obtained from Eqs. (6.2) and (6.3). The centre coordinates of (6.12) are obtained from the centre estimates of the previous iteration. The update of the cluster

centre by the CA equations involves all points in the data set without an alphacut threshold. However, the application of the OCF equations is subject to two conditions: (i) for $N_\alpha > N_{min}$ and (ii) for $u_i > \alpha_t$. For notational clarity, these two conditions and the iteration index t are omitted in the following OCF equations for the estimation of the cluster centre.

$$v_1 = \frac{B'C - BC'}{AB' - A'B} \tag{6.14}$$

$$v_2 = \frac{A'C - AC'}{A'B - AB'} \tag{6.15}$$

where

$$A = \sum_{k=1}^{N_c} (x_k - \bar{x}_k) x_k \tag{6.16}$$

$$A' = \sum_{k=1}^{N_c} (y_k - \bar{y}_k) x_k \tag{6.17}$$

$$B = \sum_{k=1}^{N_c} (x_k - \bar{x}_k) y_k \tag{6.18}$$

$$B' = \sum_{k=1}^{N_c} (y_k - \bar{y}_k) y_k \tag{6.19}$$

$$2C' = \sum_{k=1}^{N_c} (x_k - \bar{x}_k)(x_k^2 + y_k^2) \tag{6.20}$$

$$2C = \sum_{k=1}^{N_c} (y_k - \bar{y}_k)(x_k^2 + y_k^2) \tag{6.21}$$

$$\bar{x} = \frac{1}{N_c} \sum_{k=1}^{N_c} x_k \tag{6.22}$$

$$\bar{y} = \frac{1}{N_c} \sum_{k=1}^{N_c} y_k \tag{6.23}$$

The cluster radius r is computed from (6.8) using cluster centre values from (6.14) and (6.15).

6.3 Pseudo Code

A critical design feature of the algorithm is the progressive removal of good cluster points and non-cluster points, in a separate operation external to the optimisation loops for the fuzzy membership, cluster radius and centre. This strategy frees the optimisation loops to focus on the convergence of the cluster parameters, and contributes to the progressive development of clusters in two major ways. Firstly, it tests for N_{min} to overcome convergence problems and secondly, it provides the impetus (by the removal of data points) to search for a new cluster. Without the controlled removal of data points, multiple cluster development would be impossible. In addition, a beneficial spin-off from this strategy is the automatic cleaning of noisy images. The elimination of unclustered points enhances cluster

definition and thereby improves cluster detection. Furthermore, the progressive removal of cluster points also improves computation efficiency and performance.

Pseudo code to detect circular clusters.

Fix N_{min}, q, α_t and η.

Repeat

Assume $r_0 = 0$, use nearest neighbour of data centroid as v_0.

Repeat

Calculate $u_{k,t}$ from (6.5) and r_t from (6.8) if $N_\alpha >= N_{min}$ else from (6.2) and (6.3).

Calculate $\mathbf{v}_k$ from (6.9) and (6.10).

If ($N_\alpha >= N_{min}$) **Then** Calculate $\mathbf{v}_k$ from (6.14) and (6.15) and r from (6.8).

Until ($\|\mathbf{v}_{t+1}-\mathbf{v}_t\| < v_{tol}$).

If ($0 \le N_\alpha < N_{min}$) **Then** Remove N_α and update N_c.

If ($N_\alpha >= N_{min}$ and $\xi < r_{tol}$) **Then** Save cluster points N_α and update N_c.

Until ($N_c < N_{min}$).

Note: v_{tol} is a small value to control the stopping condition and t is an iteration index.

The following define the meanings of the symbols: α_t is the alphacut threshold on membership; $\mathbf{v}$ is the cluster centre; v_{tol} is the current cluster centre tolerance; N_α is the number of points in the cluster; N_c is the current count of data points; N_{min} is the minimum points per cluster specification; q is the metric exponent; r_{tol} is the radial tolerance specification; ξ is the cluster radial tolerance. For this application, $v_{tol} = 0.005$ and $r_{tol} = 0.05$ were used. The cluster radial tolerance is defined as

$$\xi = \frac{3}{N_\alpha}\sigma \tag{6.24}$$

where the cluster variance is given by

$$\sigma = \sqrt{\frac{1}{N_\alpha}\sum_{k=1}^{N_c}\left(\sqrt{(x_k - v_{1,t})^2 + (y_k - v_{2,t})^2} - r_t\right)^2} \tag{6.25}$$

Cluster centre coordinates $v_{1,t}$ and $v_{2,t}$ are obtained from OCF equations (6.14) and (6.15), and r_t from (6.8). Note the particular form of (6.24) to represent the three sigma limits of the cluster radius per cluster point. This form is used rather than the usual three sigma limits to improve screening of unwanted clusters. The algorithm has a moderate tendency to generate small size clusters (with less than 10 points) together with good clusters. The good clusters typically contain significantly higher number of points. Equation (6.24) is

found to be effective in isolating small clusters without impeding the development of clusters from a low N_{min} specification. An example of the implications of Eq. (6.24) will be given in Section 6.5. Practical considerations and the type of applications determine the specification value of r_{tol}.

6.4 Parameter Selection

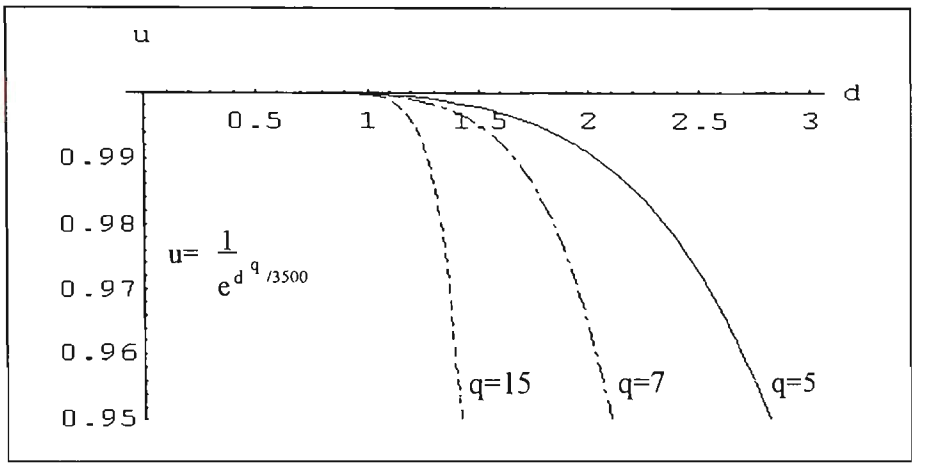
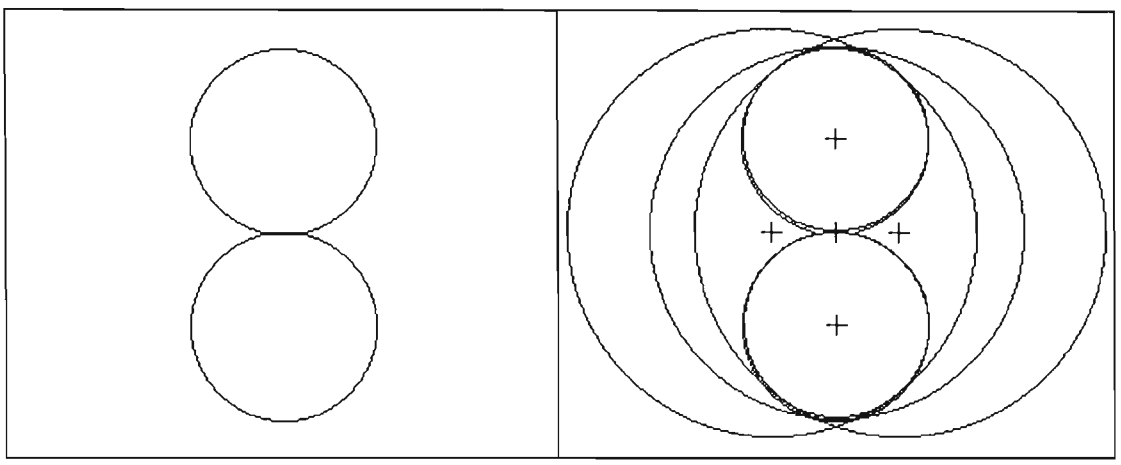


Figure 6.1 Plot of membership versus radial standard deviation for several q.

Considering the complexity of the clustering task, the significant cluster parameters of the algorithm are surprisingly few in number. The algorithm's performance is governed by two parameters, q and N_{min}, that could be preset in most cases. Figure 6.1 shows that q limits the cluster radial standard deviation d to one standard deviation (or 1 pixel unit) for values of $q > 7$ at $\alpha_t = 0.95$. With the assumptions of $\alpha_t = 0.95$ and $\eta = 3{,}500$, our experiments indicate that a high specification of q (for $q > 7$) is generally needed to detect poorly defined clusters or clusters in a noisy environment. A clean or noiseless cluster can be detected for a low q specification (for $3 < q < 7$). The number of points in the cluster determines the size of N_{min}. In the examples, it is possible to obtain good or optimal cluster performance from $N_{min} = 5$ if the clusters are clean or well defined. For clusters in a noisy environment, a substantially higher value of N_{min} is appropriate to minimise the detection of poorly defined clusters. For clusters containing several rings of different radii with a medium level of noise, a good initial set of parameter values is with $q = 7$ and $N_{min} = 5$. In most cases the presets, q and N_{min}, will yield reasonable cluster solutions. The same presets, corresponding to a particular cluster structure, will exhibit a modest degree of insensitivity to cluster numbers and invariance to scale, rotation and translation of the cluster points. However, the distribution and level of noise can have a significant bearing on the

ability of the algorithm to replicate optimal results from the same presets. If the cluster results do not conform to expectation, the recommended procedure is to search for a solution by varying q and N_{min} in a sequential order. Although these ranges may appear excessive, in reality, more than one solution may exist within the prescribed ranges. Specific issues concerning the proper selection of these parameters are discussed in the experimental results of Section 6.5.

6.5 Experimental results



(a) (b)

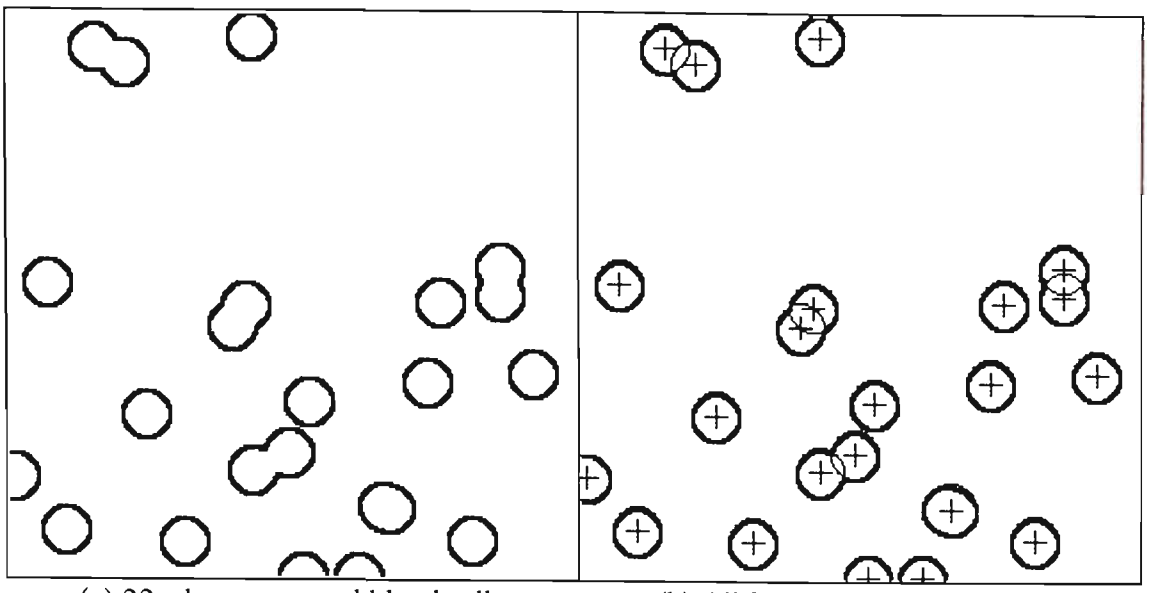
Fig. 6.2 (a) Two identical circles (538 points) vertically aligned and in contact. (b) Five possible clusters found at $q = 41$ and $N_{min} = 5$. Cluster solutions are superimposed on original unclustered rings. The two smallest cluster rings were obtained with $q = 3$ and $N_{min} = 5$.

The first example, consisting of a figure of eight formed from two identical circles, shows the influence of subjectivity on the interpretation of a cluster result. One may perhaps reasonably assume that Fig. 6.2(a) contains only two clusters. An extended cluster search discloses five rings shown in Fig. 6.2(b). The centre of each of the clusters is marked with a small plus symbol. The significance of this result suggests that it is not possible to predicate a priori the existence of particular cluster solutions at fixed values of q and N_{min}. Generally, a lower q specification tends to yield the normal cluster interpretation. The final cluster result is improved by applying equation (6.24). The results of Table 6.1 show five poorly formed clusters, numbered *1* to *5*, for $N_\alpha < 10$. In this case, it is clearly impossible to use a 3σ cluster tolerance to isolate these clusters. However, cluster validity criterion (6.24) enables the extraction of good clusters numbered *6* to *10*. The results of Table 6.1

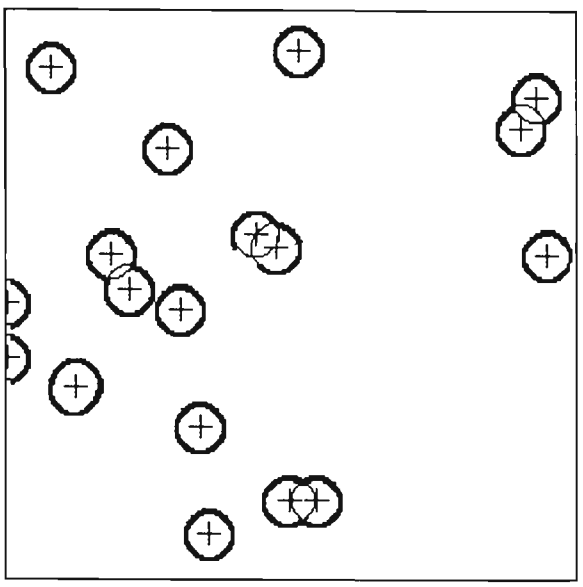
suggest accurate cluster parameter estimation. The true cluster radius and cluster centre, determined from the geometry of Fig. 6.2(a), is 47.5 pixel units. The true cluster centres are located at coordinates (141.5, 93.5) and (141.5, 188.5), corresponding to the lower and upper rings respectively.

Cluster number	N_α	v_1	v_2	Cluster radius	3σ	$3\sigma/N_\alpha$
1	8	141.5	141.5	1.2	1.333	0.1666
2	8	137.5	141.5	1.2	1.333	0.1666
3	8	145.5	141.5	1.2	1.333	0.1666
4	8	133.5	141.5	1.2	1.333	0.1666
5	8	149.5	141.5	1.2	1.333	0.1666
6	64	141.5	141.0	94.6	1.492	0.0233
7	42	109.4	141.0	104.4	1.070	0.0255
8	42	173.6	141.0	104.4	1.070	0.0255
9	172	141.5	188.5	47.5	0.750	0.0044
10	178	141.5	94.3	47.5	0.860	0.0048

Table 6.1 Summary of Fig. 6.2(b) cluster result. Cluster parameter constants are: $\alpha_t = 0.95$, $q = 41$, $N_{min} = 5$. The true cluster centres (for the two smallest circles) are at (141.5, 93.5) and (141.5, 188.5) and the true cluster radius is 47.5 units of pixels. N_α is the points count of each cluster. Point (v_1, v_2) is the cluster centre. Clusters numbered *1* to *5* with $N_\alpha < 10$ and radius < 2 are poorly formed. They do not give a useful cluster structure interpretation.



(a) 22 edge segmented blood cells. (b) All 22 cells correctly detected.

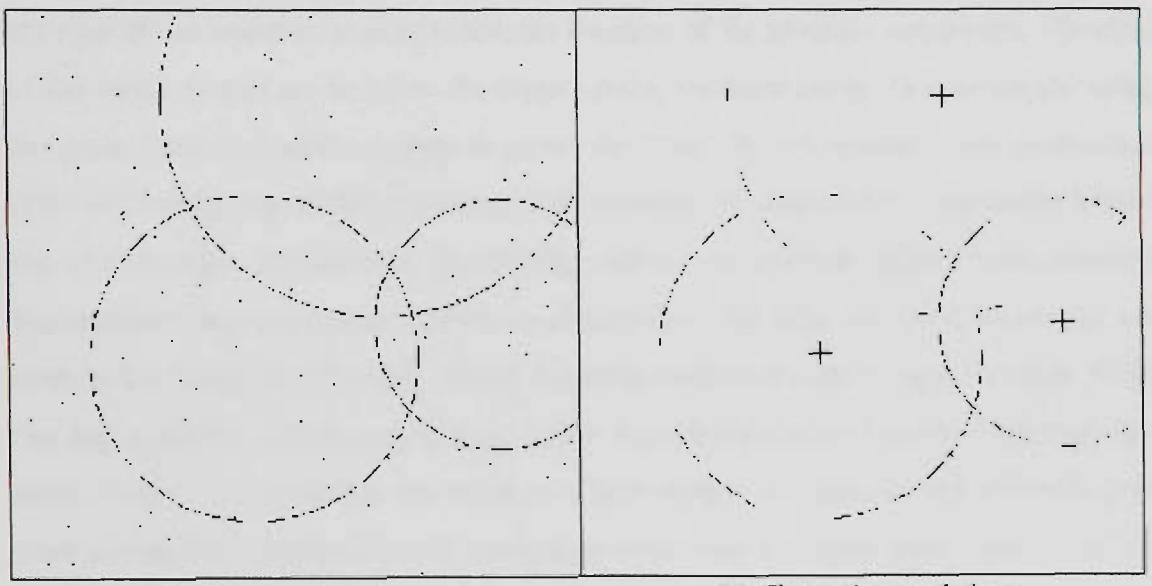


(c) All 18 cells correctly detected.

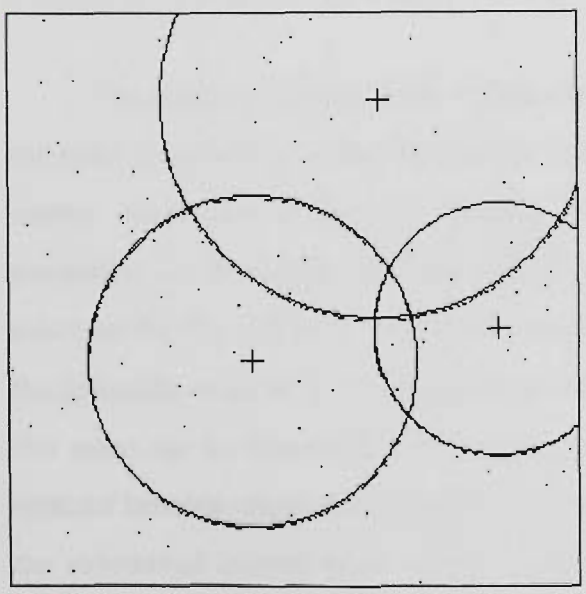
Figure 6.3 (a) Sobel segmented red blood cells comprising 3,101 points. (b) Twenty two clustered cells with $q = 7$ and $N_{min} = 5$. Image (c) is the same as (b), less four cells and rotated by 90 degrees. All eighteen cells of (c) are correctly identified for same cluster parameters as (b). Source of Fig. 6.3(a): [J.R. Parker, 1994].

The automatic inspection of blood samples is suitable for fuzzy clustering, as demonstrated in Fig. 6.3(a). The algorithm is applied to count the number of blood cells. A close inspection reveals that this is quite a complicated task because the cells not only display some irregular outlines, but also overlap to various extent. As Parker [12] noted, the partial occlusion of neighbouring cells presents some problems for conventional image processing. Although it is possible to solve this problem with the Hough transform method, the fuzzy clustering approach seems more elegant and efficient. The algorithm performed well on two different blood samples, correctly identifying all cells in Figs. 6.3(b) and

6.3(c). All 22 cells in the image of Fig. 6.3(a), consisting of 3,101 points, were detected in 15.5 seconds on an i486DX/33-MHz Intel microprocessor. The result is shown in Fig. 6.3(b), where the identified cluster is marked by a small cross at the centre of the cell. Figure 6.3(c) shows a different pattern presented to the algorithm, obtained by rotating Fig. 3(b) 90 deg. and removing four cells. Using the same parameter values as for Fig. 3(b), the algorithm counted all the eighteen cells of Fig. 3(c) correctly. The results of Figs. 3(b) and 3(c) demonstrate that the algorithm can automatically find the optimum cluster numbers from preset cluster parameter values.



(a) Three noisy rings. (b) Three clustered rings.



(c) Clustered rings of (b) superimposed on (a).

Figure 6.4 A noisy image of three rings, comprising 571 points. Image (b) shows three clusters detected with $q = 12$ and $N_{min} = 5$. Image (c) shows the detected rings superimposed on image (a). Note the accurate centres and radii of the detected cluster parameters in Fig. 6.4(c).

To evaluate clustering performance in the presence of noise, the algorithm was presented with the image of Fig. 6.4(a), consisting of fragmented outlines of three intersecting circles in a noisy background. The algorithm successfully detected all three clusters with accurate centres and radii estimation in Fig. 6.4(b). The accurate cluster centres and radii depicted in Fig. 6.4(c) demonstrate the robust clustering performance.

The next example is concerned with the automatic assembly of an industrial grinder. Figure 6.5 (a), (b) and (c) depict some randomly oriented holding positions of three different sizes of the armature housing before the insertion of the armature component. The tasks of the vision system are to locate the object centre, the inner circle for component mating and outer circle to enable a gripper to secure the object for subassembly. One pronounced effect of the edge segmentation process is the presence of residual noise, especially around the interior circle. Furthermore, the circular outlines are not well defined, with extensive fragmentation and extraneous features located between the inner and outer circles that can confuse the clustering algorithm. Given the concentration of patterns in the vicinity of the two major circles, discriminating these circles present difficulties for any clustering algorithm. Table 6.2 summarises the result of a first attempt to obtain cluster solutions from fixed parameters. Cluster solutions (consisting of the inner and outer annuli and centre) are limited to $40 \leq N_{min} \leq 49$ at unit intervals, with $q = 12$.

The cluster results of Table 6.2 indicate that the noise features in close proximity to the inner circle of Fig. 6.5(a) distorts the inner cluster circle, producing a circle that is eccentric. Elimination of the noise features, shown in Fig. 6.5(d), produces a dramatic improvement, confirming the problem diagnosis. Table 6.2 shows the transformation from no solutions for Fig. 6.5(a) to 10 solutions for Fig. 6.5(d). This pleasing result demonstrates the feasibility of using fixed parameter presets for optimal clustering. In practice, the interior noise can be minimised by controlling the lighting environment or by increasing the contrast between image and background to improve the edge segmentation process. Despite the substantial interior noise in Fig. 6.5(a), optimal cluster numbers were obtained as shown in Fig. 6.5(e). The accurate location of the four cluster rings in Fig. 6.5(f) demonstrates good clustering performance from the algorithm.

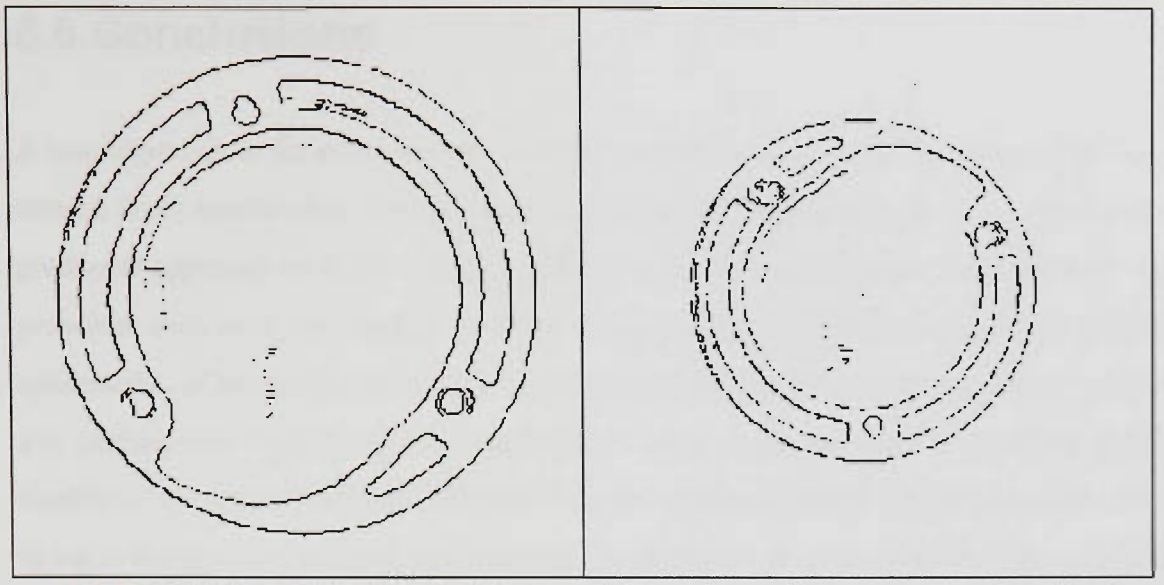
Image	N_α	Solutions for $N_{\min}$									
		40	41	42	43	44	45	46	47	48	49
Fig. 6.5a	2126	no	no	no	no	no	no	no	no	no	no
Fig. 6.5b	1186	yes	yes	yes	yes	no	no	no	yes	yes	no
Fig. 6.5c	2235	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
Fig. 6.5d	2027	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes

Table 6.2 Summary of cluster solutions for $40 \leq N_{min} \leq 49$ at unit intervals, with $q = 12$. N_α is the number of points in the cluster. $N_{\min}$ is the minimum points per cluster.

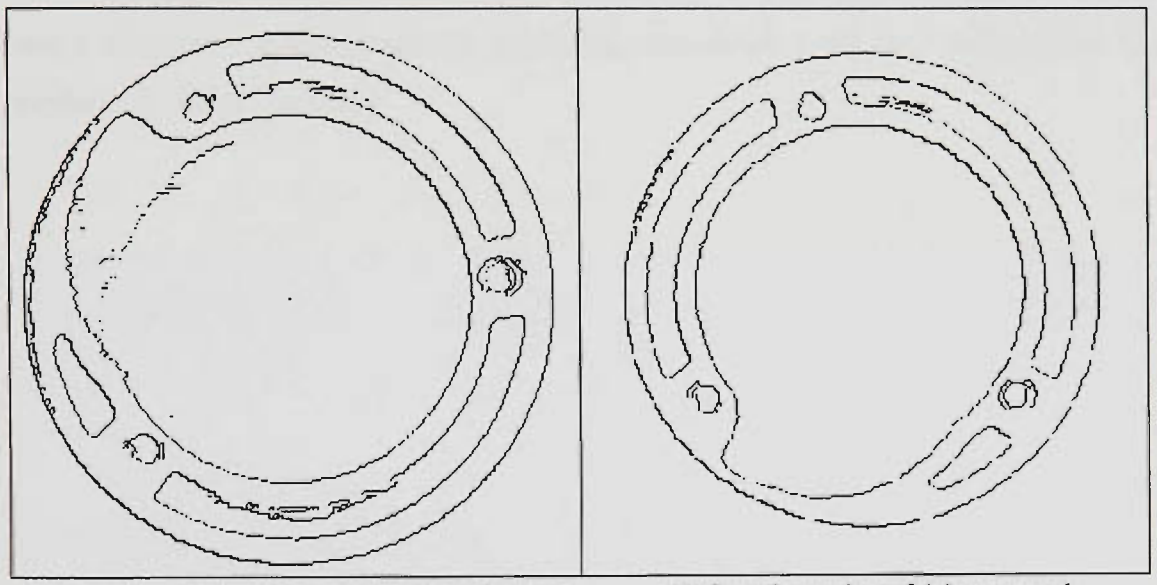
A summary of the circle statistics of Fig. 6.5(e) is given in Table 6.3, relating to the number of points in each cluster, circle centres, radii and the r_{tol} values.

Cluster number	Number of points	Centre	Radius	r_{tol}
1	299	127.49, 125.89	83.93	.0054
2	626	126.33, 107.38	107.38	.0024
3	398	127.56, 126.09	97.10	.0039
4	347	127.31, 126.65	74.33	.0040

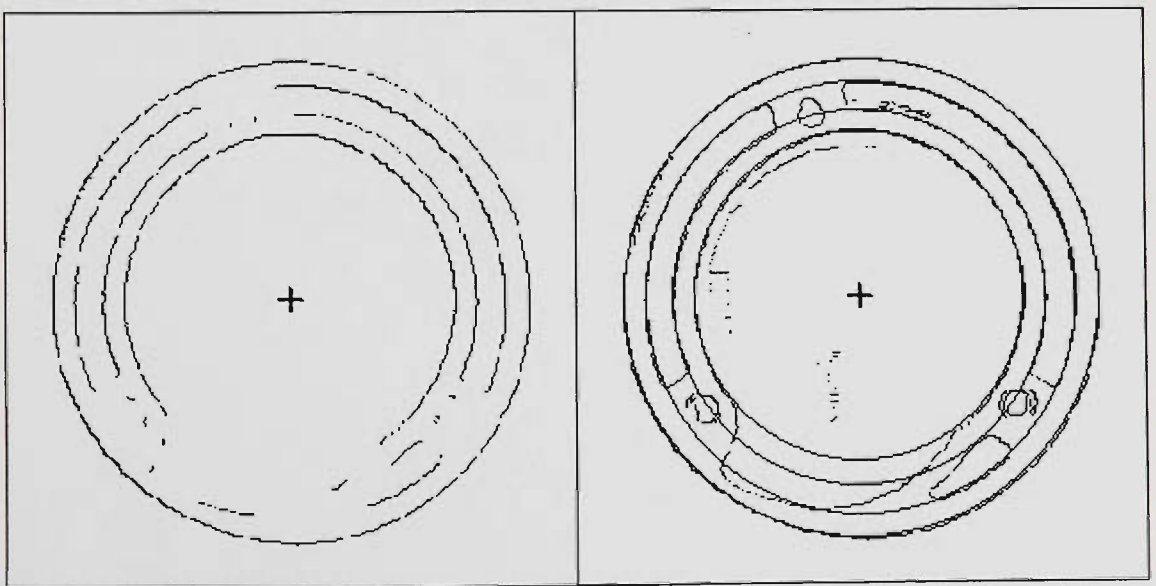
Table 6.3 Summary of Fig. 6.5(e) circle statistics. Note: Circle centre is marked by a small cross in Figs 6.5(e) and 6.5(f). The origin of the points is at the lower left corner. All dimensions are in pixel unit.



(a) Outline of armature housing top. (b) Another outline of armature housing top.



(c) Another outline of armature housing top. (d) Interior noise of (a) removed.



(e) Four segmented clusters of (a). (f) Image (a) with the four rings of (e).

Figure 6.5 Segmented outlines of an armature housing top, acquired in different orientations and positions. (e) Clusters parameters of (e) are: q=12 and N_{min}=112.

6.6 Conclusions

A new approach to fuzzy clustering based on a supervised synthesis of independently optimised fuzzy membership, cluster radius and cluster centre has been presented. This compositional approach to fuzzy clustering offers certain advantages over the analytical approaches, such as more freedom to define memberships and greater flexibility to enforce optimisation of the cluster radius and centre without the analytic constraints. Some distinctive performance features of the algorithm are (i) automatic detection of optimum cluster numbers, (ii) generalisation of detection from constant parameters and (iii) accurate clustering in the presence of noise. Additionally, the algorithm provides an alternative approach to solve analytically intractable clustering problems. The construction of separate optimisation of cluster parameters and the supervising framework entail more design effort, but produces accurate clustering.

Chapter 7

A General Pattern Recognition Method

The general pattern recognition of any arbitrary object feature is currently still very difficult to achieve primarily because of the large number of possible variations of the same feature and insufficient understanding of the nature of pattern interpretation. Nevertheless, by constraining the conditions by which the image is acquired, with a few simplifying assumptions such as a relatively low noise environment, undistorted and well-defined feature views in a 2D image plane, it is possible to obtain a satisfactory result for pattern recognition.

This chapter presents a method for the general pattern recognition of a local feature on the top face of an electric motor armature housing, consisting of the following processes: (i) image acquisition in Section 7.1, (ii) image thresholding in Section 7.2, (iii) image segmentation in Section 7.3, (iv) edge detection in Section 7.4 and (v) pattern matching in Section 7.5. This example illustrates how to use the different fuzzy clustering methods examined in chapters 4 and 6, with some of the above mentioned techniques, to construct a solution for general pattern recognition. A schematic of the pattern recognition method is shown in Fig. 7.1.

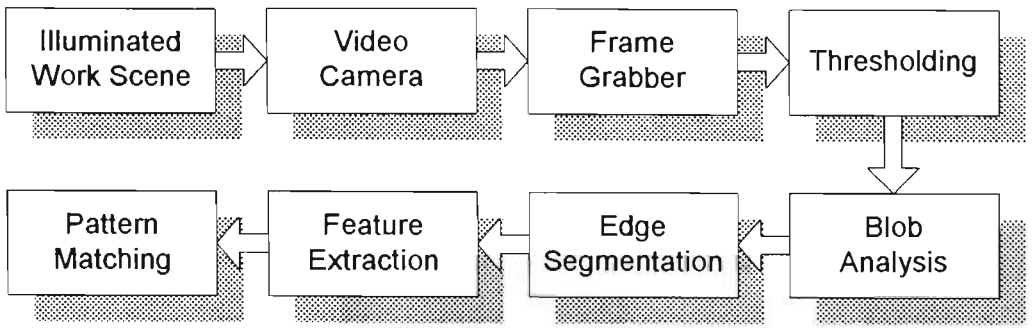


Figure 7.1 A schematic of the general pattern recognition method.

7.1 Image Acquisition

The basic requirements for image acquisition are a good lighting environment, suitable fixtures for lighting and for the placement of the objects for image capture, a video camera and a frame grabber.

Lighting involves the two important factors; quality and direction. The quality of lighting relates to the adequacy of illumination intensity to delineate object features from the background. Usually some control on illumination intensity is necessary to avoid unwanted effects of over or under exposure. Diffuse or even lighting is generally desirable to prevent casting shadows, but difficult to achieve in practice. For this experiment, four 100 watts blue tinted globes were mounted on a fixture, which has adjustable height and lighting direction to simulate diffuse lighting.

The armature housing object has a stable bottom position, simplifying the object placement for image acquisition. The dark colored object was placed on a white background to provide a good contrast. The camera was mounted on a height adjustable fixture with a cantilever to position the camera lens directly above the object (to minimise perspective distortion).

A calibration of the camera for image position and screen resolution may be needed to obtain true views of the object. Image position calibration was achieved using a calibrated grid of dot patterns (6 mm diameter dots, spaced 50 mm apart) and acquiring the dot patterns at various camera heights. Errors in the x and y axes were found to be less than $\pm$ 1 pixel unit over a length of 300 mm, and less than $\pm$ 2 pixel units over a length of 200 mm, respectively. An example of the calibration chart for the x axis image position is shown in

Fig. 7.2. The camera used is a high resolution CCD color camera, Pulnix model TMC-76 fitted with an f/1.4, 10 mm lens without auto-focussing. It has a resolution of 756(H) × 581(V) which is incompatible with the Data Translation frame grabber (model DT2871) display resolution of 512 × 512. Consequently, screen image is not displayed in true view and appears stretched along the y-axis. Using the camera calibration charts for the x and y axes, the correction factor on the y-axis was calculated to be 0.7049.

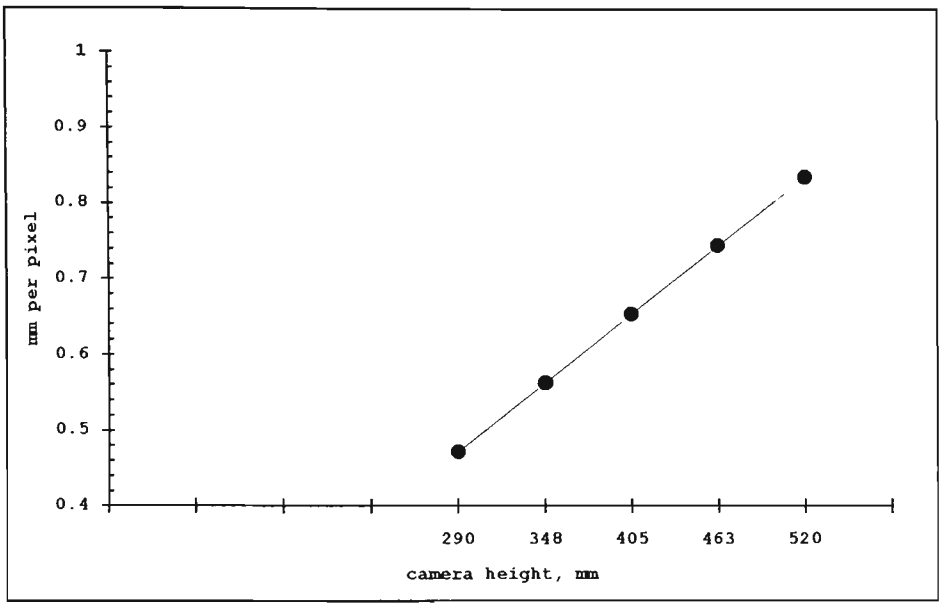
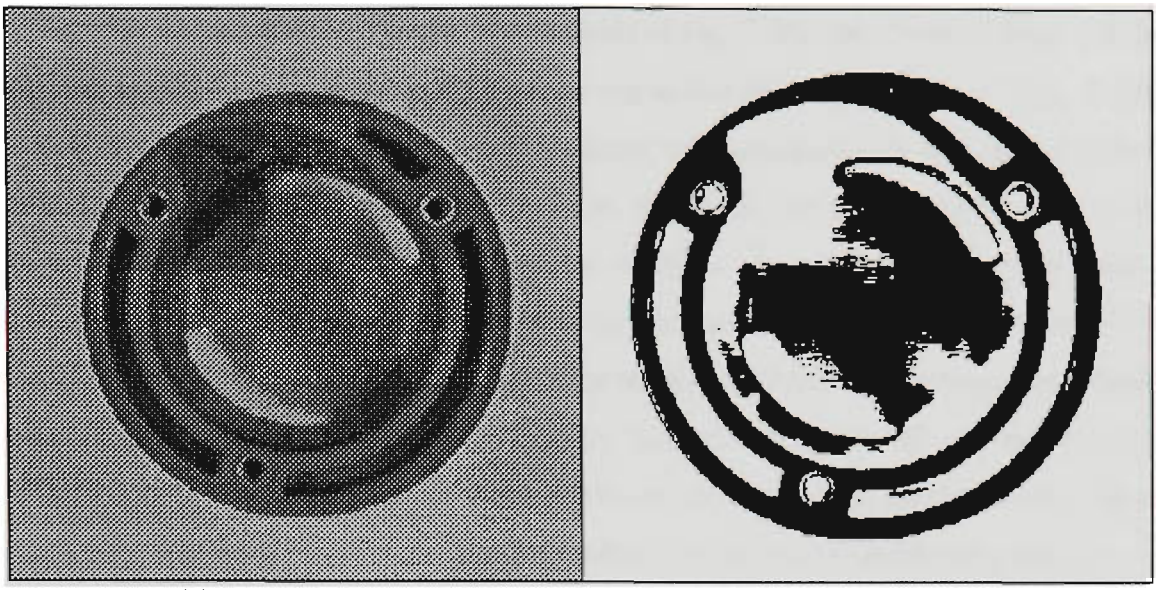


Figure 7.2 Image scale for x axis pixels.

7.2 Image Thresholding

To simplify the image processing task, the image is acquired in gray scales of 256 levels. This image is thresholded to prepare the image for blob analysis and pattern matching stages. There are many different methods for image thresholding [see Gonzalez and Woods, 1992; Haralick and Shapiro, 1992; Davies, 1990]. The fuzzy method presented in Chapter 4 can also be used effectively. However, some experimental trials are initially necessary to establish a good constant for the η adjustment equation (4.5). Once this is completed, the algorithm is straightforward to use with good discrimination under a wide range of illumination conditions.



(a) Intensity image. (b) Thresholded image of (a).

Figure 7.3 Top views of an armature housing.

Figure 7.3(a) shows the intensity image of the housing top which is only displayed in 16 gray levels, because of a limitation of the word processor. The image contained 256 levels and was thresholded as 256 levels. Figure 7.3(b) shows the binary image resulting from thresholding. Note that the binary image of Fig. 7.3(b) contains extraneous data in the centre region of the image. For our particular method of pattern recognition, the data will not adversely affect the end result, although it may present problems with other methods used. However, since the data is not useful for downstream processes, it should be removed in the interest of improved computation efficiency. The object features are isolated from the background by a method called image segmentation.

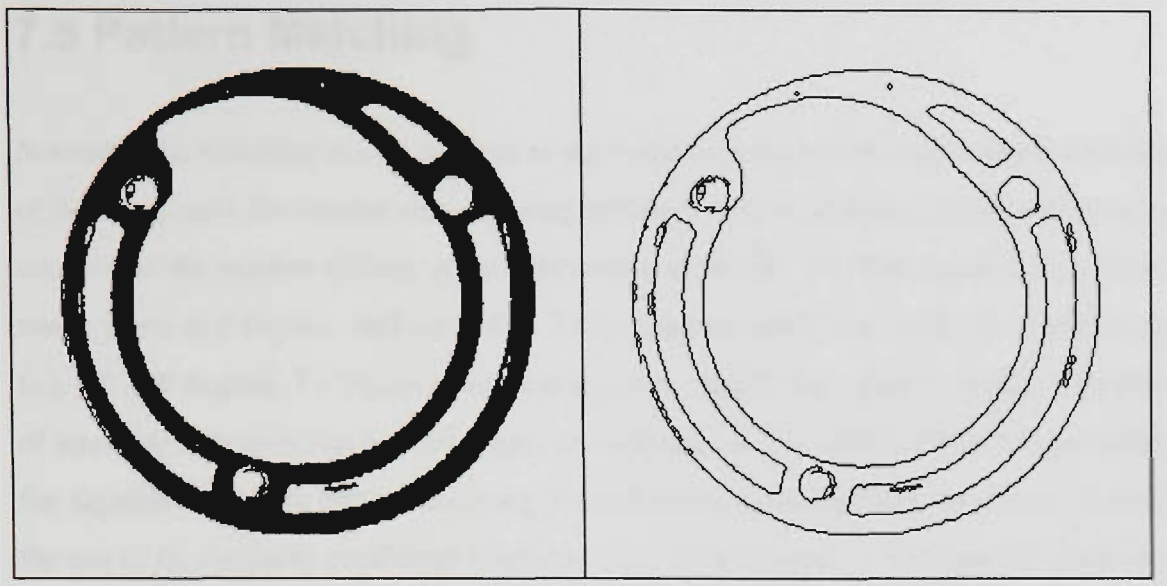
7.3 Image Segmentation

Like thresholding, there are numerous methods for image segmentation. Some examples are found in [Gonzalez and Woods, 1992; Haralick and Shapiro, 1992; Davies, 1990; Cohen, 1993; Fu and Mui, 1981]. The method we used is known as blob (or connectivity) analysis introduced by [Cunningham, 1981]. We modified his method to include some refinements such as improved segmentation accuracy of complicated blob structures and automatic edge segmentation [Im, 1992].

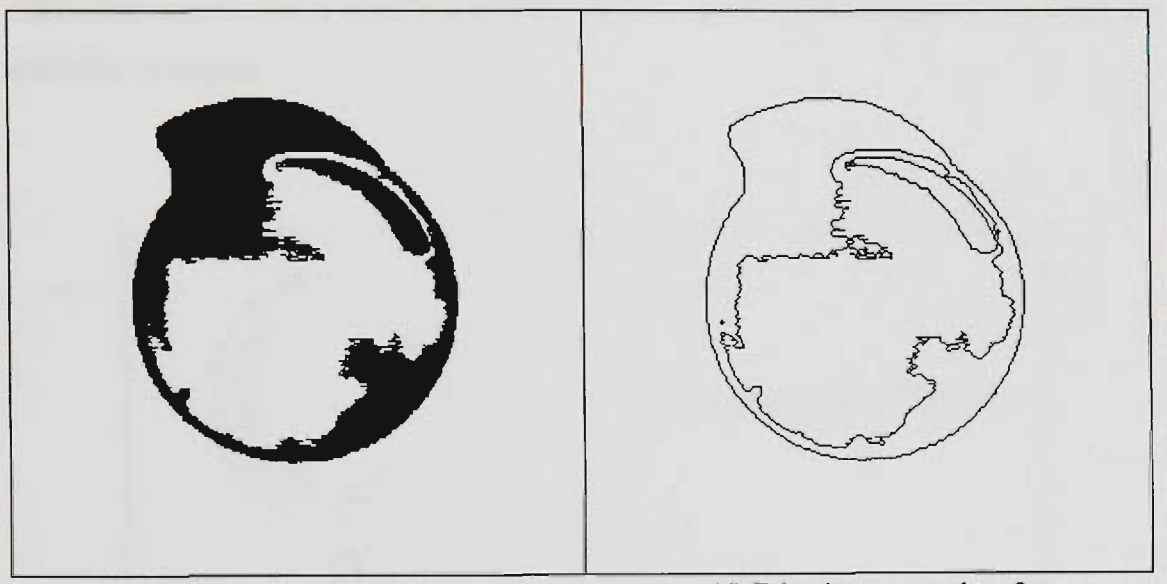
Three blobs obtained from a blob analysis of Fig. 7.3(b) are shown in Figs. 7.4 (a), (c) and (e). The corresponding edges of the segmented blobs are shown in Figs. 7.4(b), 7.4(d) and 7.4(f). For the purpose of visualisation, the segmented blob is shown as a black object in a white background. As this example illustrates, one or more blobs may be obtained from thresholding an image. Therefore, a method to identify the segmented blobs is necessary to ensure the correct blob is used for the pattern matching stage. Because blob analysis performs a normal raster scan of the image, it is easy to implement an efficient procedure within blob analysis to quantify the geometric properties of an object; for example, areas, perimeters, first and second moments of area, bounding dimensions, aspect ratio and other shape parameters. These properties may be used to identify the object.

7.4 Feature Extraction

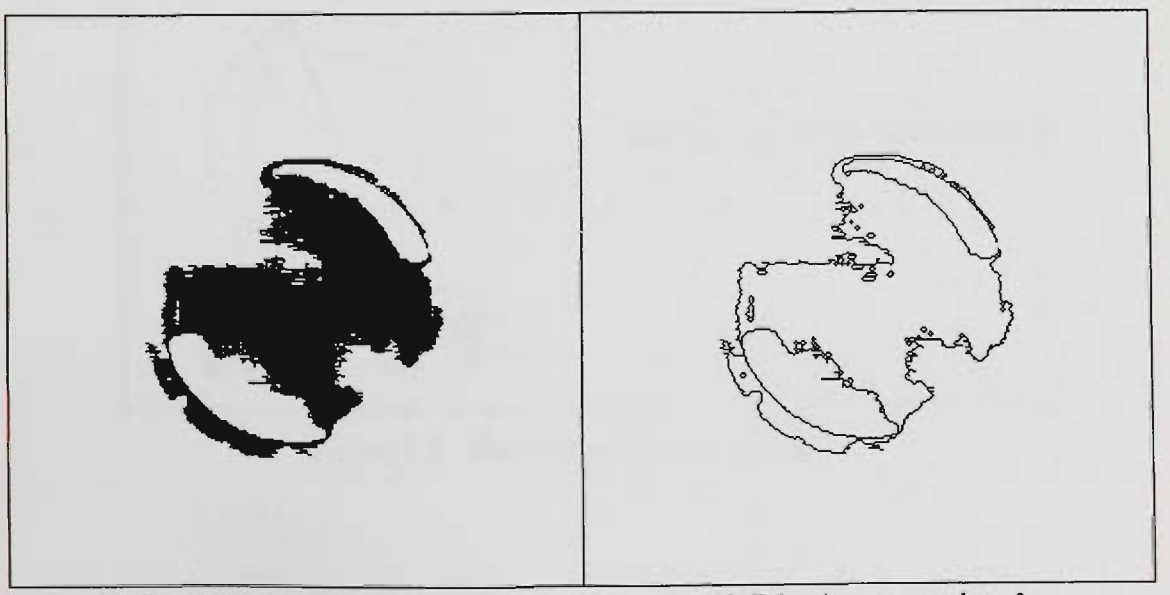
Feature extraction is used to obtain a model feature for pattern matching and also to enable a comparison of selected data features to be made against the desired model feature, during a search for a pattern match. Several silhouette based techniques, such as curvature scale space representation of points in [Mokhtarian, 1995], a parametric approach for matching of polygonal profiles [Ventura et al., 1995] and the correlation coefficient method [Im, 1992], have been used. Our method uses an array of 30 (max) mean radial profile points, where each point corresponds to a degree of rotation angle. The profile points are used to calculate the similarity coefficient (see Section 7.5.1) for pattern matching. The maximum array size is introduced to limit the processing time. The feature extraction process, illustrated in Fig. 7.5, shows the extraction of a notch profile from a window with boundaries defined by a radial angle range of 30 degrees and the inner and outer radii of 80 and 117 pixel units, respectively.



(a) Segmented image number *1*. (b) Edge image number *1*.



(c) Segmented image number *2*. (d) Edge image number *2*.



(e) Segmented image number *3*. (f) Edge image number *3*.

Figure 7.4 Three blobs extracted from Fig. 7.3(b).

7.5 Pattern Matching

Normally, the matching of data features to the model features is done over a restricted zone of the image area for reasons of processing efficiency. In our method, pattern matching is confined to the window defined in the lower drawing of Fig. 7.5. The search for a pattern match starts at 0 degrees, defined in Fig. 7.6 (a), and proceeds in a clockwise direction up to a full 360 degrees. To improve the accuracy of the search for a pattern match, a number of screening strategies can be used. These are discussed in [Im, 1992]. For this experiment, the algorithm used for pattern matching has efficient processing techniques derived from the use of (i) similarity coefficient (Section 7.5.1, from [Flusser, 1995]) and (ii) data sectoring (Section 7.5.2). Consequently, the pattern matching algorithm can implement an exhaustive search over a zone bounded by the two specified radii without any need for screening strategies.

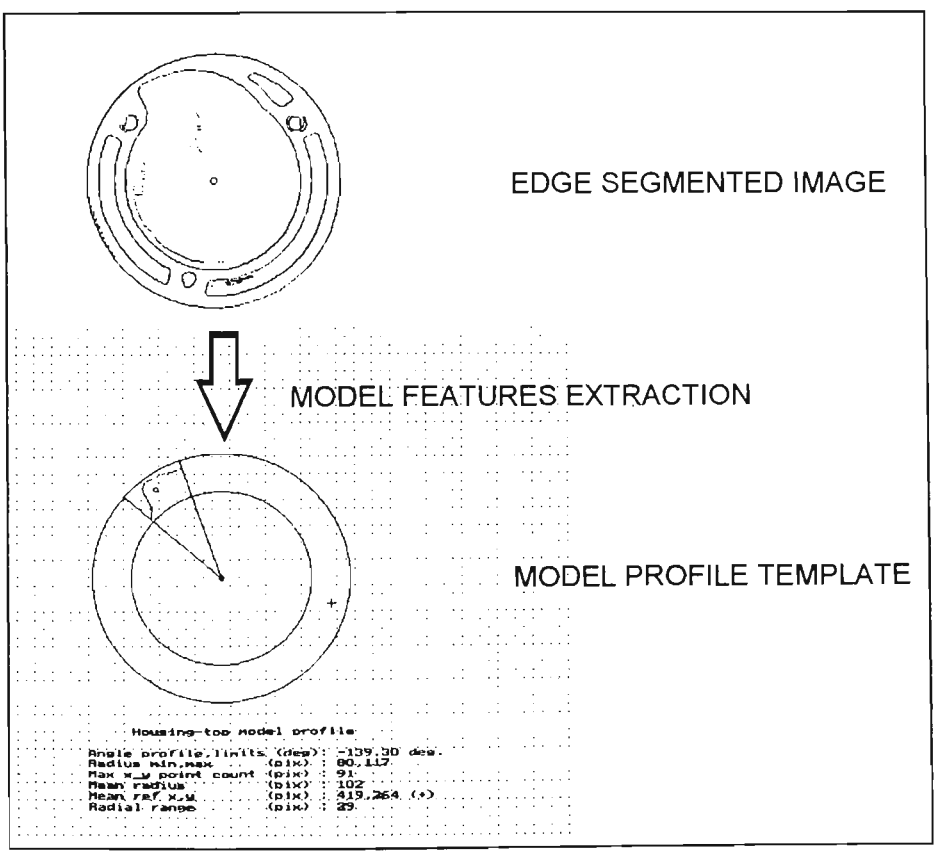


Figure 7.5. The feature extraction process.

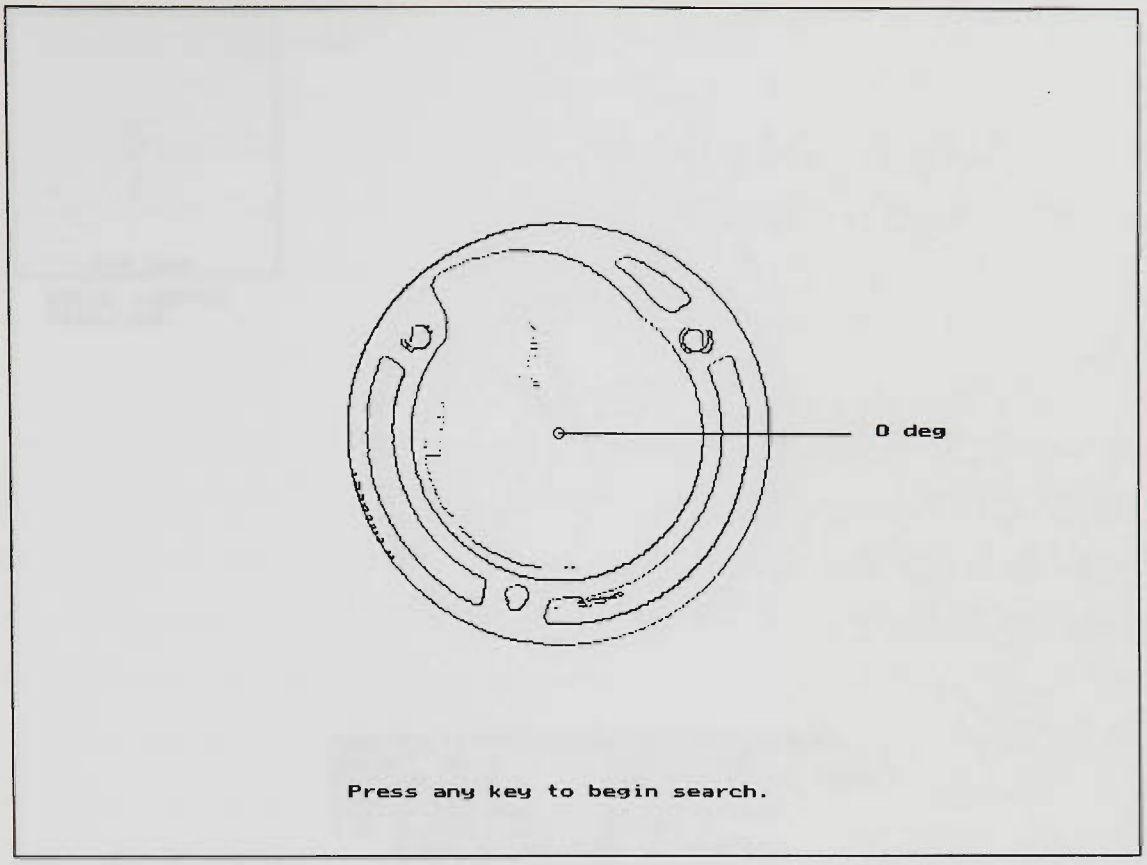


Figure 7.6 Object presented for pattern matching.

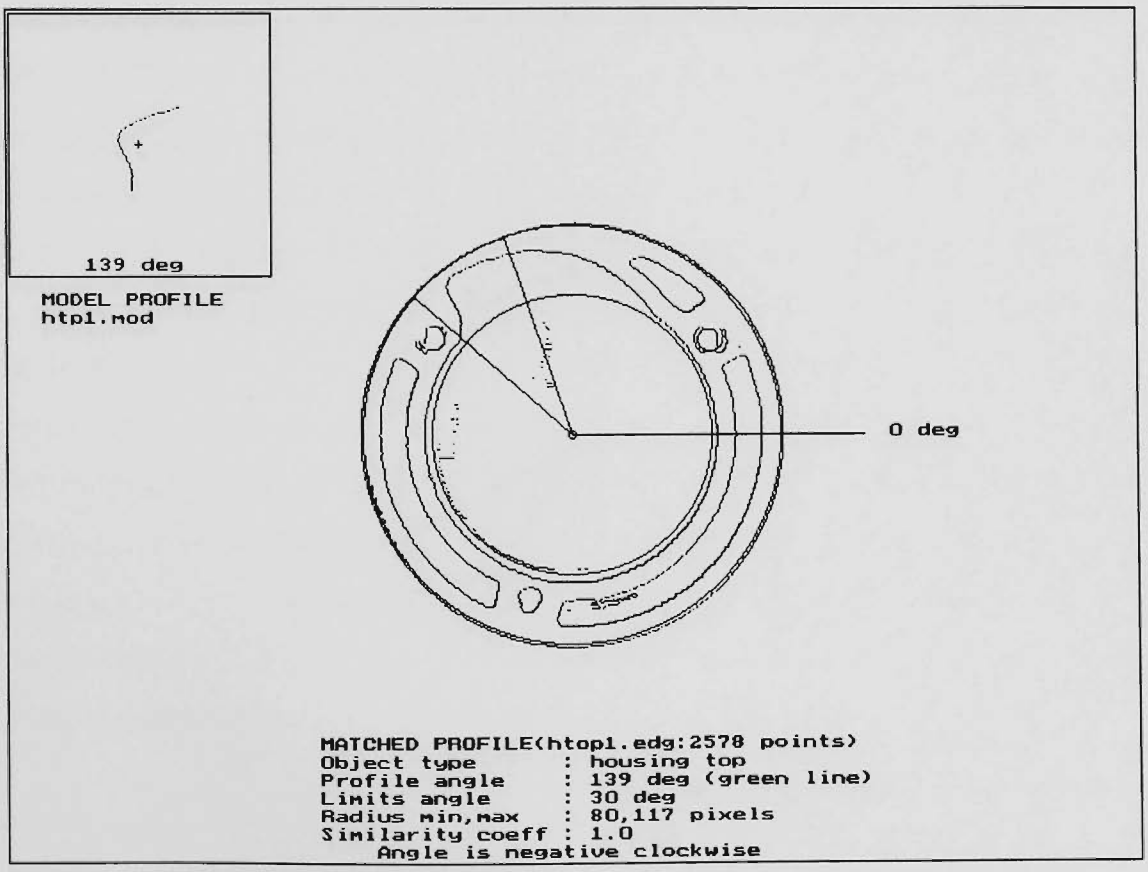


Figure 7.7 Pattern matching of a local feature at 139 degrees.

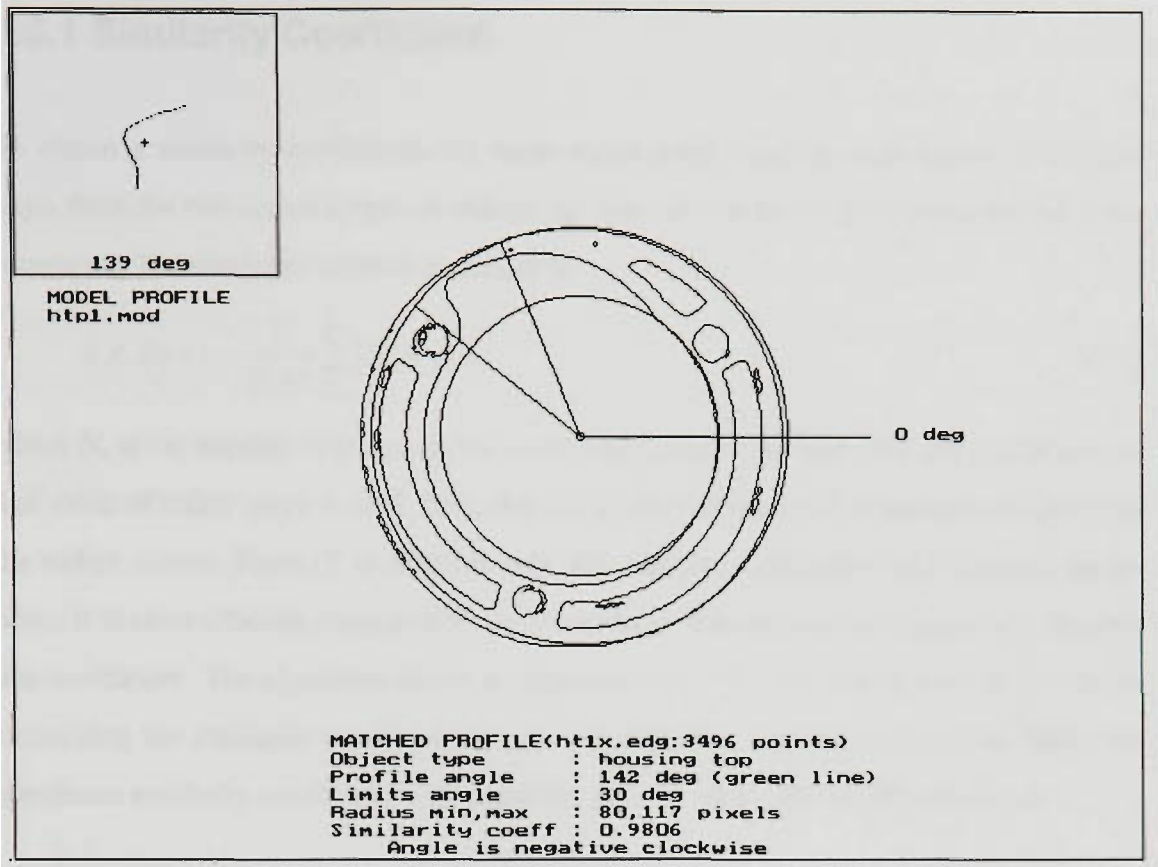


Figure 7.8 Pattern matching of a local feature at 142 degrees.

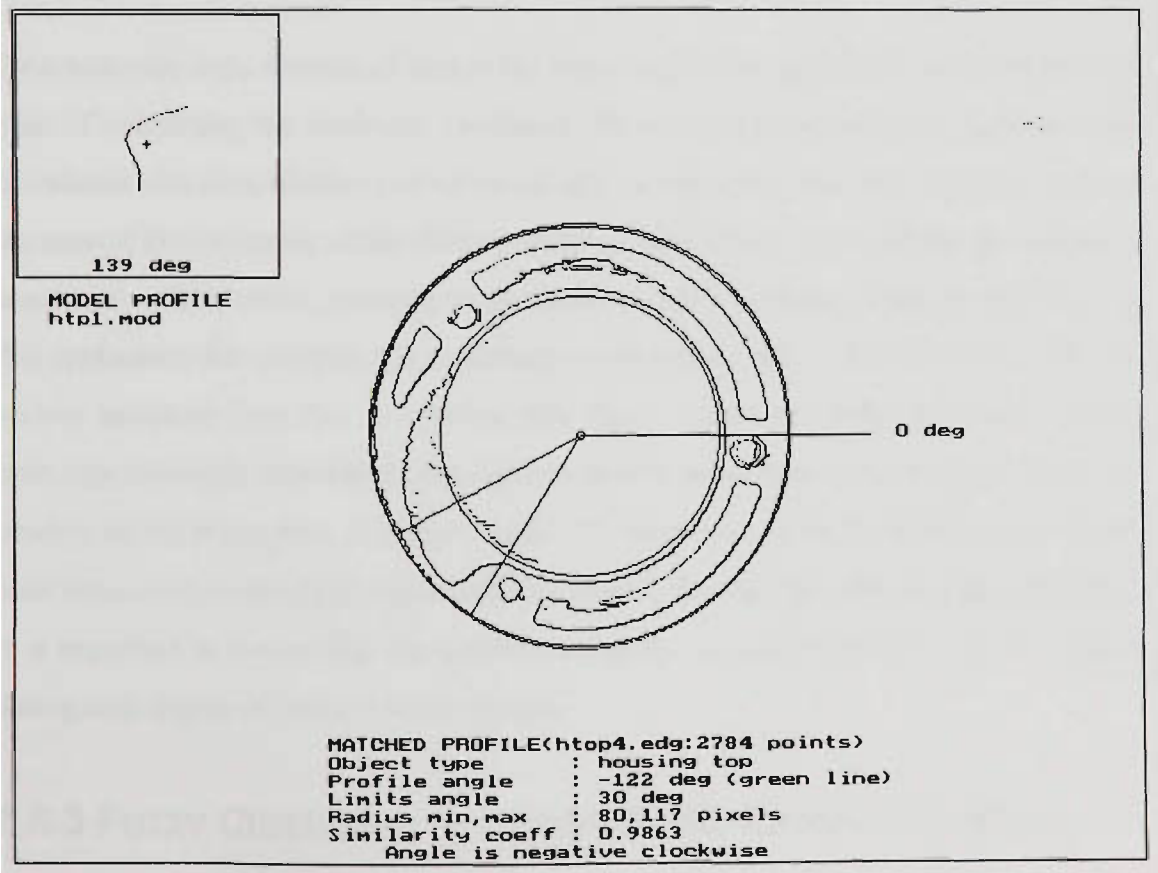


Figure 7.9 Pattern matching of a local feature at -122 degrees.

7.5.1 Similarity Coefficient

To obtain a similarity coefficient, the mean radial point value at each degree of rotation angle from the two sorted arrays of objects A_k (from the model) and B_k (from the data) are compared. The similarity relation is defined by

$$s(A,B) = 1 - \frac{1}{N_c M} \sum_{k=1}^{N_c} |A_k - B_k| \tag{7.1}$$

where N_c is the number of points in the array (maximum = 30) and M is the maximum radial value of either array A or B. Note that each value of the array is measured relative to the radius centre. Since (7.1) involves only the addition, subtraction and division operations, it is more efficient compared to other similarity indices such as a statistical correlation coefficient. The algorithm allows a minimum of $N_c = 5$ and a maximum of $N_c = 30$ in calculating the similarity coefficient $s(A,B)$. An additional procedure is used to detect the maximum similarity coefficient $s(A,B)$ and the angle at which this coefficient occurs.

7.5.2 Data Sectoring

Data sectoring is the division of data in the image space into equal sector areas for the purpose of calculating the similarity coefficient. Since the pattern match is confined to the boundaries of a data window (within which data is extracted), it is only necessary to limit the zone of data sectoring within the two specified radii. Data extracted from the window is associated with its sector, resulting in significantly improved data processing efficiency. In this application for example, the processing speed increased by a factor of five when the sectors increased from four to eighteen (this figure varies with different images). It has been experimentally established that eighteen sectors gives near optimal performance, depending on the image data. A higher number of sectors may result in sub-optimal performance because of increased processing for the sectors. In using the data sectoring technique, it is important to ensure that the sectors overlap the extreme boundaries of the window during each degree of rotation of the window.

7.5.3 Fuzzy Clustering to Locate Circle Centre and Radii

The accuracy of pattern matching depends on an accurate reference centre for the window. This is demonstrated by comparing Figs. 7.7 and 7.8. Figure 7.7 locates the optimal match

angle at 139 degrees. Note the model profile obtained from the same object at the upper left corner of Fig. 7.7). Figure 7.6 shows the same object presented for pattern matching. The orientation angle of the object in Fig. 7.8 is the same as Fig. 7.7, except that the centre of the object of Fig. 7.8 is located by a centroid of area method, a less accurate method compared to fuzzy clustering. Consequently eccentric circles formed by the rotating window are clearly visible in Fig. 7.8, because of inaccurate centring. The pattern match algorithm is however quite robust, finding a match at 142 degrees for Fig. 7.8 with an error of 3 degrees compared to the optimum match angle at 139 degrees of Fig. 7.7. The error is also reflected in a lower similarity coefficient. The fuzzy clustering method for circular cluster detection, examined in Chapter 6, gives accurate centre values needed by the pattern match algorithm. For example, Fig. 6.5 demonstrates that accurate circle centre, and the inner and outer radii of the housing rings can be obtained from the clustering method. Figure 7.9 shows another example of correct pattern match at a rotation angle of -122 degrees.

7.6 Clustering Performance

Table 7.1 summarises the processing times to detect the selected pattern at various indicated orientations of the housing top face. The processing times were measured on an i486DX/33-MHz Intel microprocessor. The processing time to match all the different profile angles, averages about 3 seconds. A major portion of the processing time was due to sectoring data into the 18 sectors. The time to compute similarity coefficients was negligible. The different orientations of the selected profile on the housing top face were correctly identified by the pattern matching algorithm, without errors.

Image	Number of points	Profile angle (deg)	Processing time (sec)
HTOP1.EDG	2578	139	3.0
HTOP2.EDG	2703	66	3.2
HTOP3.EDG	2775	-32	3.5
HTOP4.EDG	2784	-122	3.3

Table 7.1 Summary of processing times. The profile angle is measured positive in a counter clockwise direction. 18 data sectors were used in the algorithm. All four different orientations of the housing were correctly detected. The processing time was measured on an i486DX/33-MHz Intel microprocessor.

7.7 Conclusions

A general pattern recognition of an arbitrary object feature is generally difficult to accomplish successfully without simplifying assumptions that constrain the conditions by which the image features are to be detected. One such approach involves pattern matching of local feature points using similarity coefficients and a number of low level image processing routines to convert the data into a form suitable for this task. In constructing a general pattern recognition method, the fuzzy clustering methods such as those discussed in Chapter 4 for region segmentation or thresholding, and in Chapter 6 for the detection of circles have been found to be particularly useful. One of the significant benefits of fuzzy clustering methods is the accurate detection of graphic primitives such as lines, arcs and circles. Quite often, this aspect of image processing is difficult to achieve by conventional means, especially if the environment is noisy or the image is fragmented or partly occluded.

This example has demonstrated that the fuzzy clustering methods can significantly improve or simplify the pattern recognition problem. For example, the automatic detection of the inner and outer annuli of the housing top eliminates the need for extensive trials to define the probable window boundaries for the pattern matching algorithm. Moreover, the accurate detection of the annuli centre improves the detection accuracy of the optimal angle of match compared to conventional methods such as the centroid of area method.

Chapter 8

A Neural Network Approach

Neural networks open new possibilities for pattern recognition. One of the recognisable advantage of a neural network, among others, is its generalisation characteristic. In practical terms, this means a neural network, under suitable conditions of the data and training, can perform complex mapping that approximates to the desired output even though the input pattern is not exactly identical to the training pattern. The generalisation capability is typically obtained at a significant cost, involving extensive training of the network to learn the desired pattern features. Often elaborate data preparation is necessary because the form of the input data has a significant impact in the learning capability of the network. This chapter examines various ways of extending generalisation in a neural network.

Three different applications of the fuzzy neural networks based on the backpropagation paradigm are presented. Section 8.1 presents a scaled fuzzy prototype mapping method to improve object identification under a range of illumination conditions. Section 8.2 examines the design of a neural network configuration to improve object recognition. This involves training a cascaded neural network with membership functions to map a complex correlation coefficient parameter needed for pattern matching. In Section 8.3, a fuzzy neural network is implemented to improve cluster substructure identification by training a neural network to map the cluster prototype from the cluster membership function.

8.1 Improving Object Recognition with Scaled Fuzzy Prototypes

Two neural network methods for the pattern classification of real objects are examined. The Direct Mapping (DM) method defined in Section 8.1.2, uses normalised histograms for network inputs while the Scaled Fuzzy Prototype Mapping (SFPM) method, defined in Section 8.1.3, provides Effective Width Ratios (EWR) data for network inputs. The EWR defined in Section 8.1.4 is normalised intensity data scaled from FCM prototypes. The SFPM method proves to be more effective in mitigating the illumination effects compared to the DM method, thus enabling more robust cluster identification in a noisy environment.

8.1.1 Introduction

Images acquired by a camera are susceptible to illumination or reflectance variations which can have a significant impact on the results of subsequent image analysis. For example, the detection of specific colour tones in an object can be critically dependent on a controlled lighting environment that may be difficult or too costly to provide in certain environments. Consequently, there is a strong motivation to devise illumination insensitive methods for image processing.

In the DM method, input data to the neural network are in the form of normalised histograms. The histograms represent an image of 256×256 resolution with 256 gray levels of intensity. A significant advantage of histogram based image analysis is its simplicity. Its compact feature representation and rotation invariance are readily exploited without recourse to complex analysis.

8.1.2 Direct Mapping (DM) Method

DM involves mapping of 256 elements of the histogram into the five designated output classes by a backpropagation neural network, a paradigm [Rumelhart et al., 1986] that has been widely and successfully applied to pattern recognition problems since its inception in 1986. Other more advanced variants basically improve the reliability and speed of training times. The standard backpropagation model is also quite suitable for accurate classification because it can be implemented to give value data (discussed in Section 8.1.5) that indicate

a controllable degree of classification. The choice of the particular network model is not critical for the particular application.

A three-layer architecture (256:5:5, corresponding to the input, middle and output layers) was selected for the network, as shown in Figs. 8.1 and 8.2. For input data compatibility, a floating point data type was selected for each of the three layers. Whilst the DM method is simple to implement, it will be apparent later that the classifier performance is rather sensitive to illumination factors.

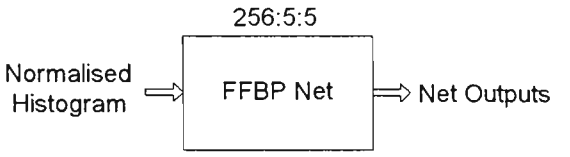


Figure 8.1 Network for DM method.

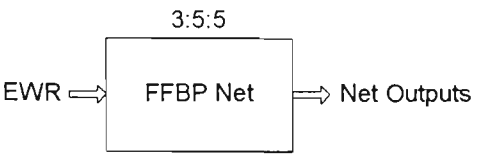


Figure 8.2 Network for SFPM method.

8.1.3 Scaled Fuzzy Prototype Mapping (SFPM) Method

The SFPM method uses the FCM algorithm [Bezdek, 1981] to generate the scaled fuzzy prototypes from the intensity data for use as input to a backpropagation neural network. The network structure also consists of three layers (3:5:5) as indicated in Fig. 8.2, except that the input layer has three processing elements corresponding to the number of the fuzzy prototypes which are normalised as Effective Width Ratios (EWR). The FCM partitions a finite set of feature vectors in real d-dimensional space R^d into c clusters or natural groups, where $1 < c < N$ is an integer. The $c \times N$ matrix $U = [u_{ik}]$ contains the fuzzy c partitions of X which satisfies the following three conditions:

$$\sum_{i=1}^{c} u_{ik} = 1, \text{ for all } k; \sum_{k=1}^{N} u_{ik} > 0, \text{ for all } i \text{ and } u_{ik} \in [0,1], \text{ for all } i,k . \tag{8.1.1}$$

The FCM computes the clusters by iterative minimisation of the general objective function

$$J_m(U,V) = \sum_{k=1}^{N}\sum_{i=1}^{c} u_{ik}^m \|\mathbf{x}_k - \mathbf{v}_i\|_A^2 \qquad \text{for } 1 \le m < \infty \tag{8.1.2}$$

where U is the fuzzy c partition of the set of cluster centres $\mathbf{v}_i \in R^d$, $\|\cdot\|_A$ the weighted inner product norm and $m \in (1,\infty)$ is the weighting exponent of the fuzzy membership. The norm is the Euclidean distance between the feature vectors and the cluster centres. Hard

clusters in X exist for $m = 1$. For $m > 1$, a local minimum of J_m exists, for all i, k and $\mathbf{x}_k \neq \mathbf{v}_j$, and for all i respectively, when

$$u_{ik} = \left[\sum_{k=1}^{c} \left(\frac{\|\mathbf{x}_k - \mathbf{v}_i\|_A}{\|\mathbf{x}_k - \mathbf{v}_j\|_A} \right)^{2/(m-1)} \right]^{-1} \tag{8.1.3}$$

$$\mathbf{v}_i = \frac{\sum_{k=1}^{N} \mathrm{u}_{ik}^{m} \mathbf{x}_k}{\sum_{k=1}^{N} \mathrm{u}_{ik}^{m}} \tag{8.1.4}$$

The exponent of the fuzzy memberships m determines the shape of the fuzzy membership. An exponent of value 2 is generally satisfactory for clustering. The algorithm proceeds with initial random values for U_0 to produce a series of values for v_i and u_{ik}. Substitutions involving u_{ik} in (8.1.3) and v_i in (8.1.4) are made repeatedly until the value of either variable converges to an acceptable limit. This algorithm is quite computation intensive. To alleviate this problem, neural network solutions were adopted. Bezdek has implemented a Feed Forward Backpropagation Cascade Correlation network with improved learning rates in [Hall et al., 1992]. It is also possible to train a backpropagation neural network to model the cluster centres (or prototypes) of the FCM algorithm (Section 8.3).

8.1.4 Effective Width Ratios (EWR)

If a histogram is used directly in training a neural network, accuracy of pattern classification may be adversely affected by lighting conditions since these factors influence the shape of the histogram. The effects of illumination variation on histogram are illustrated in Figs. 8.4(a) and 8.4(b). To improve the recognition capability of the network, it is possible to derive parameters from fuzzy clusters that are largely insensitive to illumination effects. One simple method is to scale the fuzzy prototypes as *Effective Width Ratios* (EWR),

$$w_i = \frac{c_i - b}{r} \tag{8.1.5}$$

where c_i is the cluster centre for $i = 1,2,..,N$. The symbol b denotes the lower end intensity point of the histogram above the pixel count threshold limit T, and r the profile range. The threshold T defines the new histogram profile to which pixels are considered to belong, if

they exceed T. The symbols of T, b and r are defined in Figure 8.3. Equation (8.1.5) is equivalent to a width normalisation of the histogram data.

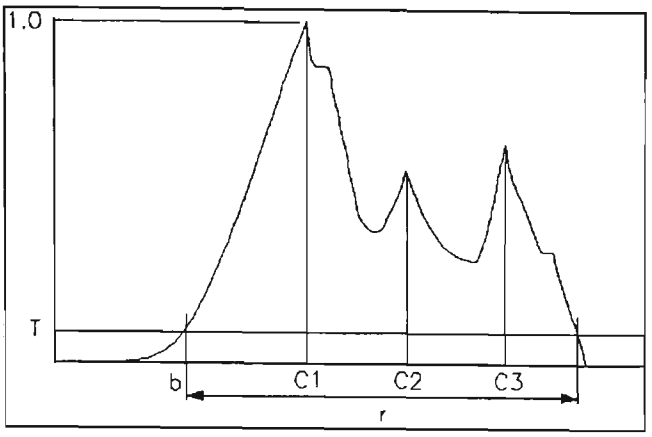
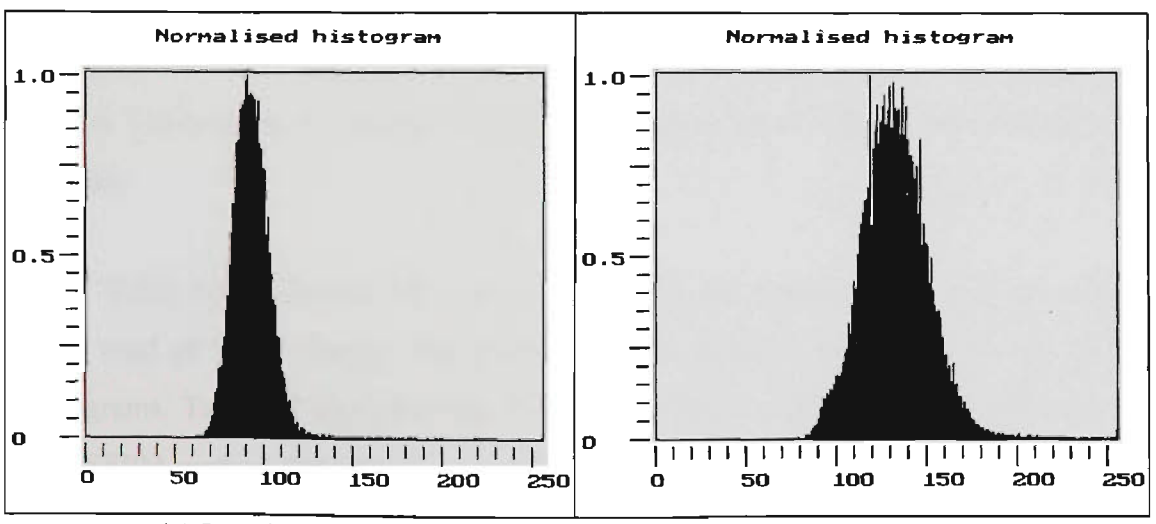


Figure 8.3 Illustration of EWR definitions.

To illustrate, an object with 3 cluster prototypes $c_1 = 113$, $c_2 = 134$ and $c_3 = 156$ has an effective intensity interval from 84 to 251 and a range $r = 167$ for T specified at 0.1 %. With $b = 84$, w_1, w_2 and w_3 compute to 0.1737, 0.2994 and 0.4311 respectively. Table 8.1 shows the EWR for 3, 4 and 5 clusters of the same object at high and low levels of illumination. The low and high illumination levels have histogram patterns shown in Fig. 8.4(a) and Fig. 8.4(b) respectively. Despite the significant difference in illumination intensity, the good agreement in the EWR for each corresponding cluster confirms the substantial validity of (8.1.5). It is observed from Table 8.1 that (8.1.5) holds for any number of clusters produced by FCM and for any range of illumination except at the extremities. At these zones, either saturation or under exposure occurs with the loss of feature discrimination.



(a) Low illumination. (b) High illumination.

Figure 8.4 Histograms of identical object at two different levels of illumination.

Figure (8.4)	Cluster Prototypes					Effective Width Ratios				
	c_1	c_2	c_3	c_4	c_5	w_1	w_2	w_3	w_4	w_5
High I	113	134	156			.1737	.2994	.4311		
Low I	83	94	107			.1856	.2990	.4330		
High I	108	125	142	161		.1437	.2455	.3472	.4611	
Low I	80	89	98	109		.1546	.2474	.3402	.4536	
High I	105	121	134	147	165	.1257	.2216	.2994	.3772	.4850
Low I	78	86	93	101	111	.1340	.2165	.2887	.3711	.4740

Table 8.1. Comparison of EWR at low and high levels of illumination, I.

8.1.5 Experimental Results

A criterion function based on the Euclidean distance is used to determine the nearest class and to evaluate the accuracy of the network's response. The distance formula is

$$d_i = \min_j ||\mathbf{v}_i - \mathbf{v}_j|| \quad \forall i \neq j \tag{8.1.6}$$

where $\mathbf{v}_i$ and $\mathbf{v}_j$ are the EWR vectors of objects i and j, and d_i is the minimum Euclidean distance between $\mathbf{v}_i$ and $\mathbf{v}_j$ for all $i \neq j$.

The nearest class according to the minimum distance criterion of (8.1.6) is indicated under the *MD* column of Tables 8.2, 8.3 and 8.4. The network's response to the histogram inputs and the scaled fuzzy prototypes are presented in Tables 8.2 and 8.3 respectively. The five output classes are represented by the five columns under the caption of Network Test Response. Each column of the output response represents one of the nominated class corresponding to features of objects *a1* to *a5*. For example, column *a1* refers to the class of *a1* object. The class assigned by the network is indicated by highest row value, shown shaded. The network was trained with image objects *a1* to *a5*. Objects *a6* to *a10* represent test data.

Table 8.2 shows that the direct method produced 2 misclassifications (*a8* and *a10*) for a total of 5 test classes. The misclassification occurred because of the similarity of histograms. Table 8.3 gives the response of the network trained using the SFPM method. The SFPM method demonstrates a significant improvement in classification performance compared to the DM method. All test objects (*a6* to *a10*, the same as Table 8.2) were cor-

rectly classified without errors. This result is attributed to the insensitivity of EWR to the illumination factor inherent in the image histogram.

Test Class	Network Test Response					*MD*
	a1	*a2*	*a3*	*a4*	*a5*	
a6	.0000	.9981	.0009	.0006	.0028	*a2*
a7	.0076	.0030	.0054	.8793	.0956	*a4*
a8	.0024	.0018	.0181	.0013	.9603	*a4*
a9	.0021	.0021	.0160	.0015	.9796	*a5*
a10	.0031	.0046	.0008	.9853	.0174	*a3*

Table 8.2 Network response using the DM method. Symbol *MD* = minimum distance criterion of (8.1.6). Class assigned by the network is shown shaded. Note: Two misclassifications for test objects *a8* and *a10*.

Test Obj	Effective Width Ratios (EWR)			Network Test Response					*MD*
				a1	*a2*	*a3*	*a4*	*a5*	
a6	.4199	.7449	.8692	.0018	.9988	.0002	.0001	.0000	*a2*
a7	.1969	.4655	.8717	.0007	.0000	.0009	.9987	.0001	*a4*
a8	.1523	.5000	.9607	.0000	.0000	.0197	.9943	.0113	*a4*
a9	.1149	.4010	.9854	.0000	.0000	.0008	.0188	.9874	*a5*
a10	.0872	.6333	.8485	.0027	.0000	.9984	.0002	.0004	*a3*

Table 8.3 Network response using the SFPM method. Class assigned by the network is shown shaded. Note: All five test objects are correctly classified.

In Table 8.4, the test objects *a11* to *a15* were artificially created (using same w_2 and w_3 but varying w_1) to test the network's response to classes close to the decision boundary of classes *a1* and *a4*. The network predicts a decision boundary (theoretically at 0.5 value) at *a12* instead of the optimum minimum at *a13*. The boundary disparity is considered small and can be minimised with an improved training regimen and a greater number of training patterns. At the class boundary, the network produces a value approximating 0.5 for each of the nearest classes (*a1* and *a4*). This experiment demonstrates that a back-propagation neural network approximates the discriminant function of a Bayes classifier.

Test Object	Effective Width Ratios (EWR)			Network Test Response					*MD*
				a1	*a2*	*a3*	*a4*	*a5*	
a11	.1800	.4896	.8072	.1281	.0000	.0074	.4970	.0000	*a4*
a12	.1930	.4896	.8072	.4758	.0000	.0011	.4987	.0001	*a4*
a13	.2125	.4896	.8072	.8849	.0000	.0001	.4853	.0000	*a1*, *a4*
a14	.2200	.4896	.8072	.9317	.0001	.0001	.4827	.0000	*a1*
a15	.2300	.4896	.8072	.9603	.0001	.0000	.4849	.0000	*a1*

Table 8.4 Network response near a decision boundary between *a1* and *a4*. Symbol *MD* = minimum distance criterion of (8.1.6). Class or classes assigned by the network is shown shaded.

8.1.6 Conclusions

Two different methods for classifying normalised histogram data have been presented. The performance of the classifier based on the direct method was unsatisfactory because the shape of the histogram profile was affected by illumination factors. The classifier based on the SFPM method was superior because the effective width ratios proved to be effective in isolating the illumination factor from the raw image data, to enable more accurate pattern recognition.

8.2 Improving Object Recognition for Pattern Matching

The design of a neural network configuration for object recognition is described. Recognition is achieved by pattern matching of a local profile using correlation coefficient. Supervised networks configured as a conventional classifier and three variations of a fuzzy classifier are investigated. Their performances are compared with the statistical correlation coefficient used as the reference. The relative percentage error of the correlation coefficients from the fuzzy neural network design was significantly less than the results from the conventional neural network design, indicating improved accuracy from fuzzy networks. Both the fuzzy and non-fuzzy networks produced the correct angle of match.

8.2.1 Introduction

Object recognition is a high level image processing task preceded by the basic downstream tasks such as image acquisition, preprocessing, segmentation and feature extraction. Three categories of object recognition may be identified: (1) decision-theoretic, (2) structural (or syntactic) and (3) image interpretation [Woods and Gonzalez, 1992]. The method adopted belongs to the decision-theoretic category. The pose of an object, defined as the angular orientation in a 2D plan view, is determined with a combination of classical image preprocessing methods and finally optimised with a fuzzy neural classifier. The aim is to demonstrate firstly that a fuzzy classifier can be designed to match the performance of the conventional object recognition methods and secondly, a neural network, given the same input data, can generalise more accurately with fuzzy stages compared to a conventional classifier.

Numerous papers have been published over the last decade in the area of object recognition. The Neocognitron developed by [Fukushima et al., 1983] paved the way for the application of neural networks to object recognition. The results were outstanding. Modelled on the human visual system, the Neocognitron was capable of recognising complex patterns with a high degree of accuracy. Since then more progress has been made in the neural network models. Four main types exist today: (i) Hopfield's associative memory, (ii)

Kohonen's self organising maps, (iii) Carpenter and Grosberg's adaptive resonance models and (iv) Rumelhart's feed forward back propagation (FFBP).

The FFBP model is quite widely used for image processing and pattern recognition. Gosh et al. [Gosh et al., 1993] modified the structure of the FFBP so that it does not require supervised training (unlike a conventional FFBP model) for the segmentation of objects. Several fuzzy indices were used to measure the system errors. The learning rates were evaluated for different error measures.

To improve object recognition, we used a supervised FFBP configured as a fuzzifier to generate the membership functions. Each function consists of a set of 3 ordered membership grades. The network structure is shown in Fig. 8.5. The values of the membership function are used as inputs for the defuzzifier to approximate the corresponding correlation coefficient.

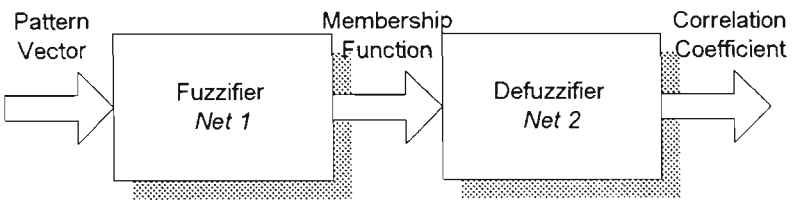
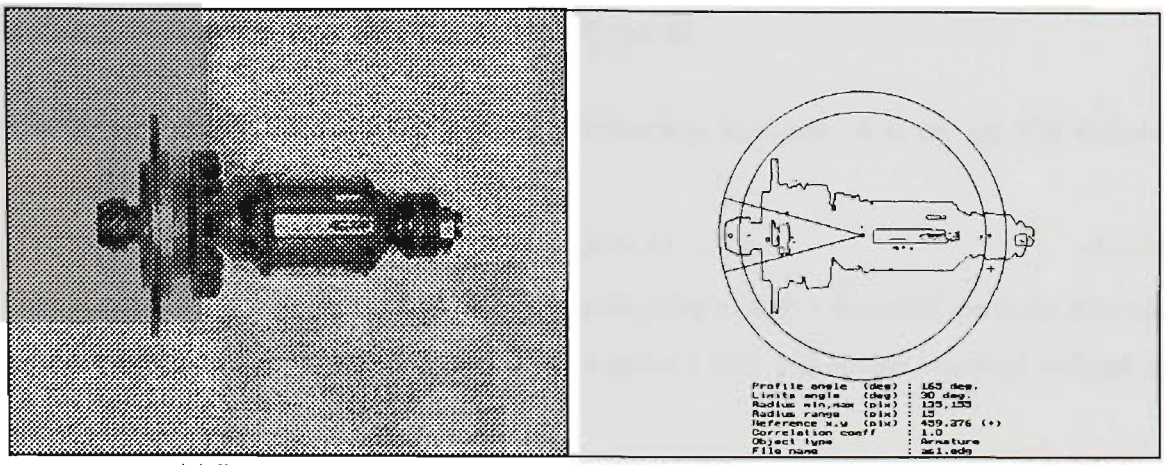


Figure 8.5 A fuzzy neural network configuration for object recognition.

8.2.2 Image Processing

A sample of an intensity image is given in Fig. 8.6(a). The image is subsequently thresholded and segmented by connectivity analysis (see Chapter 7) to produce a silhouette of the object shown in Fig. 8.6(b). Finally, a local feature of the object profile is extracted as a vector of 30 mean radial points about the object's centroid of area, at one degree intervals of rotation from the nominated profile angle. Data on the area centroid and other geometric properties such as area, moments, major axis angle and perimeter, are obtained from the blob analysis program. The feature selector described above is used to build the model profile data base and also to obtain the object pattern vector for the fuzzifier *net 1*.



(a) Intensity image. (b) Edge detected image for pattern matching.

Figure 8.6 Image of an electric motor armature.

8.2.3 Network Configurations

8.2.3.1 Conventional Design (*Type 4*)

A direct design method is to train the FFBP net with the desired set of outputs ie. the maximum correlation coefficients corresponding to different orientations of the profile from the same object. For the experiment, five pattern vectors from a local profile of the armature object at *163* to *167* degrees were selected to train the network. The angle of maximum correlation is *165* degrees which is also referred to as the profile angle. The input object data and correlation coefficients (target values) for network training are summarised below:

$ar163 = 0.8522$, $ar164 = 0.9574$, $ar165 = 1.0000$

$ar166 = 0.9580$, $ar167 = 0.8429$

The name of the object on the left side of the equality symbol represents the armature object. The suffix denotes the object's pose angle. In the above case, the angle varied from *163* to *167* degrees. The object's reference pose angle is represented by the middle angle. In the above case, the reference angle is at *165* degrees.

A FFBP network layer structure of 30:15:5 represents the 30 floating points of each input vector from a local object profile and the 5 floating points for each of the output class of coefficients. The choice for the single middle layer is guided by the need to provide sufficient weights for learning.

8.2.3.2 Fuzzy Design (*Type 2 and Type 3*)

The fuzzy design involves fuzzy sets and membership functions. A fuzzy set A is defined as a collection of ordered pairs,

$$A = \{(M_A(\mathbf{x}_i),\mathbf{x}_i),\ i = 1,2,..,N\} \text{ for } 0 \le M_A(\mathbf{x}_i) \le 1,\ \forall i \tag{8.2.1}$$

The membership function $M_A(\mathbf{x}_i)$ associates each point $\mathbf{x}_i$ with a real number in the interval [0,1]. Another useful concept is the fuzziness index I with a fuzziness measure defined in [Gosh et al., 1993] as

1. $I(A)$ = minimum $\Leftrightarrow M_A = 0$ or 1 $\forall i$.
2. $I(A)$ = maximum $\Leftrightarrow M_A = 0.5$ $\forall i$.
3. $I(A) \ge I(A^*)$ where A^* is a sharpened version of A defined as,

   $M_{A^*}(\mathbf{x}_i) \ge M_A(\mathbf{x}_i)$ if $M_A(\mathbf{x}_i) \ge 0.5$

   $M_{A^*}(\mathbf{x}_i) \le M_A(\mathbf{x}_i)$ if $M_A(\mathbf{x}_i) \le 0.5$
4. $I(A) = I(A^c)$ where A^c is the complement set of A.

For a Euclidean distance, an index of fuzzy set A having N supporting points is defined as

$$\nu(A) = \frac{2}{N}\sqrt{\sum_{i=1}^{N}\{M_A(\mathbf{x}_i) - M_{A'}(\mathbf{x}_i)\}^2} \tag{8.2.1}$$

An ordinary set is defined as

$$M_{A'}(\mathbf{x}_i) = \begin{cases} 0 & \text{if} \quad M_A(\mathbf{x}_i) \le 0.5 \\ 1 & \text{if} \quad M_A(\mathbf{x}_i) \ge 0.5 \end{cases} \tag{8.2.2}$$

The block schematic for a fuzzy neural network is shown in Fig. 8.5. In the training mode, *Net 1* was trained with the membership functions of Figs. 8.8 and 8.9 (except for the interpolated functions denoted by the faint broken lines) and *Net 2* with their corresponding coefficients as given in Table 8.6. In the production (or test response) mode, new data is presented to *Net 1* resulting in the correlation coefficient from *Net 2*. The network structure for *Net 1* and *Net 2* were 30:10:3 and 3:3:1 respectively.

(A) Hard Solution (*Type 1*)

The hard solution has $M_A(\mathbf{x}_i) = 1$ as shown in Fig. 8.7. For this particular case, the network response can be easily predicted (hence training is unnecessary for discussion purpose). The coefficients for each of the objects from *ar206* to *ar348* must be identical to the refer-

ence object (ie. *ar163* to *ar167*, theoretically). The fuzzifier assigns each of the input vectors to one of five distinct classes of the coefficients without any class overlap.

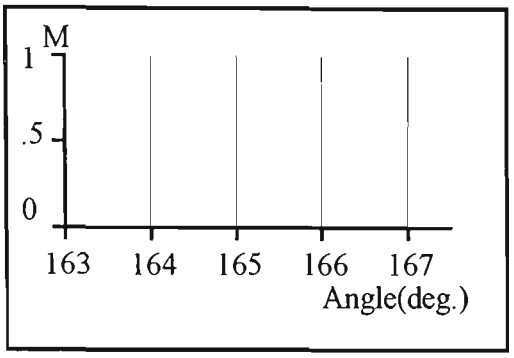


Figure 8.7 Membership functions for a crisp solution (*Type 1*).

(B) Soft Solution with Narrow Support (*Type 2*)

The soft solution provides more freedom for the design of the membership function. For simplicity, only a triangular function is considered. The support (or base) of the membership function consists of 3 points at *163*, *165* and *167* degrees with grade levels as shown in Fig. 8.8. Optimum pose is at *165* degrees. Each of the three functions has a different triangular profile. The dissimilar functions reflect the different degrees of fuzziness according to the relation of (8.2.1). For example, M_c maximised at *165* degrees is expected to be more accurate than the other functions maximised to *163* (M_a) and *167* (M_e) degrees. The fuzziness associated with the triangular functions helps to shape the neural network response to one which is non-linear and convex (bell shape). How does this response characteristic solve the pattern recognition problem? If the response is non-linear and convex, then it is possible to pick the maximum coefficient and optimum angle of match, which is impossible with a linear or monotonically increasing response characteristic. The fuzzy functions provide one effective way to coerce this type of response from a neural network.

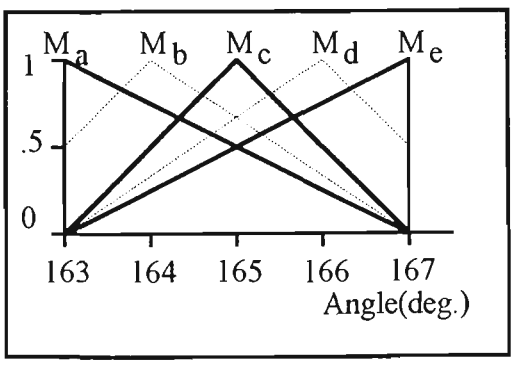


Figure 8.8 Membership functions for a soft solution with narrow supports (*Type 2*).

(C) Soft solution with Wide Support (*Type 3*)

The membership functions for this category are shown in Fig. 8.9. In contrast to the previous functions, these functions have base membership grades increased from 0 to 0.5 with a corresponding increase in the fuzziness index of Eq. (8.2.1). Theoretically, the wider support of these membership functions should produce less accurate prediction of the coefficients from the defuzzifier. In Section 8.2.4, we see that this observation is confirmed.

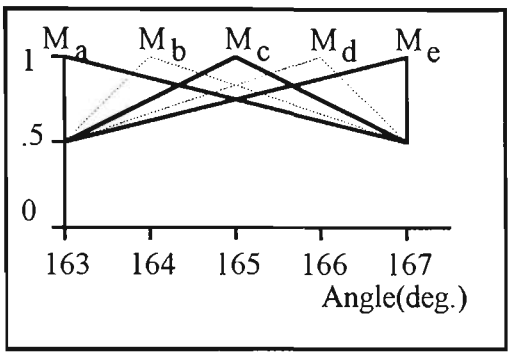


Figure 8.9 Membership functions for a soft solution with wide supports (*Type 3*).

8.2.4 Results

Results for the production mode response for *Type 2* and *Type 3* network configurations are summarised in Table 8.5. The response correlation coefficients for all four configurations and their relative error (with reference to the statistical correlation coefficient) are presented in Tables 8.6 and 8.7 respectively. The most accurate prediction, at the optimum pose angle, is obtained from the fuzzy neural network design (Types *1*, *2* and *3*). Accuracy better than 98.5 % is obtained from the crisp (*Type 1*) and soft (*Type 2*) solution with narrow support. The soft solution (*Type 3*) with a wider support yielded an accuracy only slightly worse at 97 %, but still better than the solution of the conventional (*Type 4*) design (at 92% accuracy). The results of Tables 8.6 and 8.7 confirm the validity of (8.2.1) and demonstrate the usefulness of the fuzzy functions in shaping the neural network's response. All the network configurations correctly predicted the optimum pose angle.

An advantage of the soft solution design (*Type 2*) is that less data is needed for both training and production modes, hence better performance. Moreover, intermediate points of the function can be interpolated by the fuzzy neural network with reasonable accuracy (represented by the faint broken lines of Figs. 8.8 and 8.9). The crisp fuzzy solution

(*Type 1*) and the conventional design (*Type 4*) requires a membership point at each angle of the search space for neural network training, thus longer training time.

The results of Tables 8.5, 8.6 and 8.7 suggest, with additional support from other researchers in this field [Pal, 1992], that the fuzzy neural network configuration provides more accurate predictive capability compared to a conventional design.

Object	Angle (deg)	Fuzzy Membership Function *Type 2*			Fuzzy Membership Function *Type 3*		
ar163	163	.9938	.4998	.0000	.9901	.7506	.5000
ar164	164	.5796	.9470	.0001	.8809	.9620	.3679
ar165	165	.0063	.9890	.0050	.4993	.9857	.4998
ar166	166	.0000	.9519	.5546	.3654	.9610	.8877
ar167	167	.0000	.4995	.9950	.5005	.7494	.9914
ar206	206	.9843	.4963	.0000	.9935	.5974	.6609
ar207	207	.3869	.9385	.0003	.4292	.9247	.4901
ar208	208	.0041	.9808	.0118	.6138	.9776	.5493
ar209	209	.0000	.8853	.6989	.4693	.9521	.8783
ar210	210	.0000	.2926	.9948	.6017	.7274	.9887
ar253	253	.9863	.6933	.0000	.9945	.6767	.4833
ar254	254	.2693	.9782	.0002	.9220	.9571	.3008
ar255	255	.0013	.9926	.0146	.5210	.9896	.3567
ar256	256	.0000	.9356	.8563	.3007	.9804	.7863
ar257	257	.0000	.3863	.9986	.3953	.8589	.9835
ar344	344	.9687	.3314	.0001	.9917	.6373	.6532
ar345	345	.3664	.8665	.0006	.9261	.9223	.5266
ar346	346	.0049	.9620	.0161	.7161	.9600	.6593
ar347	347	.0001	.8891	.6354	.6421	.8979	.9242
ar348	348	.0000	.3530	.9929	.7775	.5521	.9920

Table 8.5. Summary of membership functions. Note that the training data set consists of *ar163*, *ar165* and *ar167*, corresponding to membership functions M_a, M_c and M_e of Figs. 8.8 and 8.9. Each function is represented by three points, hence the three columns associated with *Type 2* and *Type 3* fuzzy membership function. The data given in *Type 2* and *Type 3* membership function columns are the responses from the neural network.

Object	Angle (deg)	Stat Coeff	Neural Network Coefficients			
			Type 1	*Type 2*	*Type 3*	*Type 4*
ar163	163	.8522	.8522	.8526	.8565	.8521
ar164	164	.9574	.9574	.9750	.9705	.9549
ar165	165	1.000	1.000	.9895	.9864	.9873
ar166	166	.9570	.9570	.9734	.9692	.9539
ar167	167	.8492	.8492	.8496	.8523	.8491
ar206	206	.8212	.8522	.7628	.6402	.8556
ar207	207	.9373	.9574	.9634	.9446	.8786
ar208	208	.9912	1.000	.9892	.9780	.9528
ar209	209	.9592	.9570	.9793	.9588	.9244
ar210	210	.8557	.8492	.8501	.7982	.7090
ar253	253	.8157	.8522	.9067	.8113	.9550
ar254	254	.9354	.9574	.9604	.9719	.8696
ar255	255	.9959	1.000	.9895	.9905	.9772
ar256	256	.9732	.9570	.9843	.9821	.9740
ar257	257	8783	.8492	.9081	.9309	.6796
ar344	344	.8215	.8522	.7929	.6896	.7554
ar345	345	.9399	.9574	.9668	.9388	.7699
a346	346	.9869	1.000	.9885	.9578	.9043
a347	347	.9540	.9570	.9754	.9072	.7779
a348	348	.8481	.8492	.7842	.5058	.5374

Table 8.6. Summary of statistical correlation coefficients. Symbols: *Type 1* = crisp solution of fuzzy network, *Type 2* = soft solution of fuzzy network with narrow support, *Type 3* = soft solution of fuzzy network with wide support and *Type 4* = conventional network classifier. Stat. Coeff. denotes the statistical correlation coefficient. The training data set consists of *ar163*, *ar165* and *ar167*. The network's response at optimum match angle is shown shaded.

Object	Angle (deg)	Stat Coeff	Relative Error (%)			
			Type 1	*Type 2*	*Type 3*	*Type 4*
ar163	163	.8522	0	+0.05	+0.50	-0.01
ar164	164	.9574	0	+1.84	+1.37	-0.26
ar165	165	1.000	0	-1.05	-1.37	-1.27
ar166	166	.9570	0	+1.71	+1.27	-0.32
ar167	167	.8492	0	+0.05	+0.37	-0.01
ar206	206	.8212	+3.78	-7.11	-22.0	+4.19
ar207	207	.9373	+2.14	+2.78	+0.78	-6.26
ar208	208	.9912	+0.89	-0.20	-1.33	-3.87
ar209	209	.9592	-0.23	+2.10	-0.04	-3.63
ar210	210	.8557	-0.76	-0.65	-6.72	-17.14
ar253	253	.8157	+4.47	+11.16	-0.54	+17.1
ar254	254	.9354	+2.35	+2.67	+3.9	-7.03
ar255	255	.9959	+0.41	-0.64	-0.54	-1.88
ar256	256	.9732	-1.66	+1.14	+0.91	+0.08
ar257	257	8783	-3.31	+3.39	+5.99	-22.6
ar344	344	.8215	+3.74	-3.48	-16.1	-7.99
ar345	345	.9399	+1.86	+2.86	-0.12	-18.1
a346	346	.9869	-1.31	+0.16	-2.93	-8.37
a347	347	.9540	-0.31	+2.24	-4.91	-18.46
a348	348	.8481	-0.13	-7.53	-40.4	-36.6

Table 8.7. Summary of relative errors. Symbols: *Type 1* = crisp solution of fuzzy network, *Type 2* = soft solution of fuzzy network with narrow support, *Type 3* = soft solution of fuzzy network with wide support and *Type 4* = conventional network classifier. Stat. Coeff. denotes the statistical correlation coefficient. The training data set consists of *ar163*, *ar165* and *ar167*. The network's response at optimum match angle is shown shaded.

8.2.5 Conclusions

Several fuzzy neural network designs to improve object recognition have been described. Using a standard statistical correlation coefficient as a reference, the performances of the different network configurations are compared for accuracy of prediction. It is demonstrated that the crisp and soft fuzzy membership functions give the most accurate result with less than 1.5 % error, compared to a conventional neural network design (with 8 % error).

8.3 Implementing a Neural Network for a Progressive Fuzzy Clustering Algorithm

A neural network based scheme for the detection of fuzzy cluster prototypes from normalised histogram of a real world image is described. These prototypes can be subsequently processed external to the neural network to enable viable real time fuzzy clustering application. The direct mapping of the histogram data, whilst simple to implement, is demonstrated to suffer from errors related to a bias condition associated with weight distribution of the network. The proposed method mitigates this problem by using a conventional back-propagation neural network with output responses trained to five points of a fuzzy membership function. Test responses from this network produced less than 5 percent error for the prototype centres.

8.3 1 Introduction

A new approach using a Feed Forward Back Propagation (FFBP) neural network to obtain cluster prototypes from the image data is presented. In contrast to the multiple cluster assignment of FCM, the FFBP is trained to recognise and respond to a single cluster prototype determined by a progressive clustering algorithm. The neural network structure is sufficiently flexible to accommodate other training regimens. The CPCM based progressive clustering avoids the practical difficulty of establishing cluster validity criteria. Furthermore, this strategy permits the adoption of various clustering models within the same network or in any compatible network paradigms without being restricted to a particular fuzzy model.

To perform progressive cluster extraction, the FFBP generates the cluster prototype or centre corresponding to the input pattern vector (256 patterns). Subsequently, external to the neural network, a clustering program uses the prototype to extract the cluster from the image data set. The remaining image data (after removal of the cluster) is re-presented to the network to generate another cluster prototype until eventually all possible clusters in the image data are exhausted.

8.3.2 A Progressive Fuzzy Clustering Algorithm

A general fuzzy objective function for a single prototype cluster development can be defined as

$$J_m(U,V) = \sum_{i=1}^{N} u_i^m f(d) \tag{8.3.1}$$

where $f(d)$ is a distance function and m is a fuzzifier constant. Membership is denoted by u_i for $i = 1,...,N$ where N is the number of data features. For an intensity histogram, N equals 256. The cluster prototype is represented by v. Previously in Section 2.4.7.2, we solved the membership of (2.4.48) by direct differentiation of the objective function (2.4.47). This membership has a similar form to the membership for (8.3.1). To illustrate an alternative procedure, we consider a solution using Lagrange multipliers. Let the sum of the memberships be a positive real number $K > 0$ with a constraint function

$$g(u) = \sum_{i=1}^{N} u_i - K \tag{8.3.2}$$

Applying the method of Lagrange multipliers to minimise J_m for membership u subject to constraint function g yields,

$$mu^{m-1} f(d) = \lambda$$

which may be reduced to

$$u_i = \frac{1}{\exp\left(d_i^2 / \eta\right)} \tag{8.3.3}$$

where

$$d_i = \|\mathbf{x}_i - \mathbf{v}\| \tag{8.3.4}$$

is the Euclidean distance of data point $\mathbf{x}_i$ from prototype $\mathbf{v}$ and the prototype is defined by

$$\mathbf{v} = \frac{\sum_{i=1}^{N} u_i^2 \mathbf{x}_i}{\sum_{i=1}^{N} u_i^2} \tag{8.3.5}$$

Equation (8.3.3) is obtained by applying the lower limit ($u = 0$) and the upper limit ($u = 1$) of membership u and taking the distance function to be $f(d) = \exp(d^2/\eta)$. The constant numerator term is assumed to be 1 for a fuzzifier value of $m = 2$. The η constant in the exponent serves as a reference level for all points in the data set. One suitable choice for η is

$$\eta = \frac{s}{N}\sum_{i=1}^{N}\|\mathbf{x}_i - \mathbf{v}\|^2 \tag{8.3.6}$$

which contains a scale factor s.

Pseudo code to perform progressive fuzzy clustering.

Fix α, N_{min}, and s.

Repeat

Assume initial $\mathbf{v}_0$ from nearest neighbour of data centroid.

Calculate η from (8.3.6).

Repeat

Calculate u_i from (8.3.3).

Calculate $\mathbf{v}_i$ from (8.3.5).

Until $\|\mathbf{v}_t - \mathbf{v}_{t-1}\| < \varepsilon$.

If $(0 \le N_\alpha < N_{min})$ **Then** Remove N_α and update N_c.

If $(N_\alpha \ge N_{min})$ **Then** Save and remove cluster points of prototype $\mathbf{v}$ and update N_c.

Until $(N_c < N_{min})$.

Note that ε is a small value to control the stopping point and t is an iteration index.

8.3.3 Neural Network Implementation

A 3-layer architecture is used for training purposes with node sizes of each layer being determined by the problem domain. The network was trained with normalised intensity histograms of digitised images, each acquired in 256 levels of grey and in resolution of 256 x 256 pixels. Data for the target prototype set were generated from the progressive fuzzy clustering algorithm.

8.3.3.1 Single Output Configuration

The design configuration for the single output network shown in Fig. 8.10 is the simplest and most direct means to train a network. Five processing elements were selected for the middle layer of the FFBP network. It will shortly be demonstrated that this network architecture suffers from significant errors of pattern recall because of a bias condition associated with the weight distribution. This error appears to be an intrinsic condition of a network paradigm that relies on pattern learning by weight adjustments.

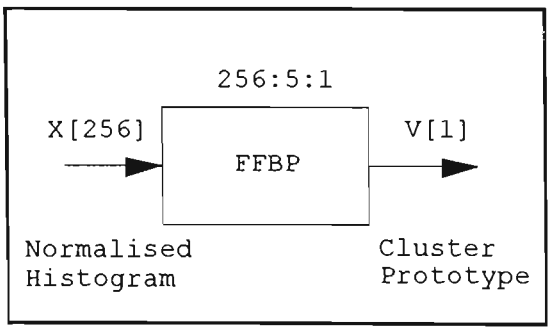


Figure 8.10 Single output neural net.

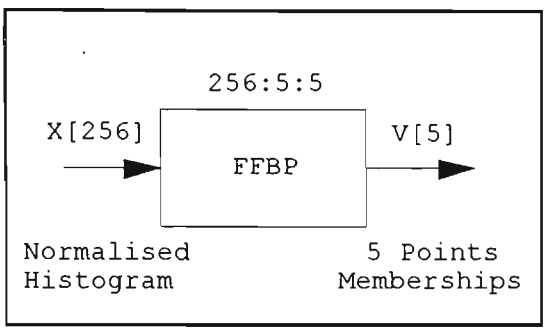


Figure 8.11. Multiple membership outputs net.

8.3.3.2 Multiple Output Configuration

To minimise the anomalies of such biases in the weight distribution, a network depicted in Fig. 8.11 with multiple outputs is proposed. Both the middle and output layers have the same number of 5 processing elements. The 5 outputs of the network correspond to the 5 points on the membership function (8.3.3), obtained by applying alphacuts at $\alpha_t = 0.2$, $\alpha_t = 0.6$ and $\alpha_t = 1$. The 5 points are obtained from the intersections of the alphacut and the membership function in a left to right order: left $\alpha_t = 0.2$, left $\alpha_t = 0.6$, centre $\alpha_t = 1.0$, right $\alpha_t = 0.6$ and right $\alpha_t = 0.2$. The 5 points of the membership function contain more structural information on the input pattern compared to a single output network. Furthermore, the five outputs help to spread possible errors among five nodes instead of one, and thus reduce the error at each node. For these reasons, the multiple output network is expected to map the input patterns more accurately, compared to the single output network. The correct output response is obtained from the output node for $\alpha_t = 1$.

8.3.4 Experimental Results

The experimental results of Table 8.8 contains a summary of the FFBP network prototype responses and the actual prototype value calculated from the progressive cluster algorithm of Section 8.3.2. Each of the test patterns *T1* to *T15* represents new patterns not used in the training of the network. The column under *Case A* refers to training with 13 pattern files with a *mean absolute error* (MAE) = (*target* – *output*) less than 0.018. For *Case B*, a training set of 10 pattern files were used with an MAE < 0.017. Both *Case A* and *Case B* refer to the multiple outputs network of Fig. 8.11. The single output network response (Fig. 8.10) is represented under the column of *Case C* for an MAE < 0.01. Despite the low MAE, two significant response errors were produced in *Case C* to test patterns *T2* and *T14*

(shaded responses in Table 8.8). As indicated in Section 8.3.3.2, this error is attributed to the peculiar biases of the weight distribution of the network's weight matrix. In other words, there is a learning problem due to cross-talk [Kosko, 1992]. *Case B* (with 10 training patterns) represents a network design (multiple outputs configuration) that attempts to minimise the learning problem of *Case C* predictive errors. At first sight, *Case B* appears to be worst than *Case C* because of response errors associated with test patterns *T4*, *T7*, *T8*, *T13* and *T14* (shaded responses in Table 8.8). However a closer scrutiny reveals that *Case B* errors occur where the network has no closely matched pattern exemplars from the training phase for values centred near 0.2 and 0.5. The inclusion of additional training patterns to cover these gaps in *Case A* (with 13 training patterns) produces the best result with errors less than 5 %. Considering the few set of training patterns used in this experiment, the results of the prototype response from the multiple output FFBP neural network (Fig. 8.11) is quite good and could be improved with more training patterns.

Test Pattern	Actual Response	*Case A* Response	*Case B* Response	*Case C* Response
T1	0.4077	0.4625	0.4473	0.4909
T2	0.5310	0.5365	0.4642	0.7078
T3	0.6221	0.6391	0.6384	0.6526
T4	0.5540	0.5076	0.4344	0.5753
T5	0.6770	0.5970	0.6763	0.5959
T6	0.3188	0.2845	0.4452	0.3154
T7	0.2504	0.2747	0.6191	0.2800
T8	0.2767	0.2337	0.5947	0.2470
T9	0.3971	0.4084	0.3909	0.3836
T10	0.4118	0.4444	0.4695	0.4070
T11	0.4307	0.4370	0.4269	0.4356
T12	0.4346	0.5060	0.4226	0.5562
T13	0.5438	0.5136	0.4346	0.5143
T14	0.7392	0.7450	0.5740	0.5883
T15	0.8002	0.8028	0.8070	0.8937

Table 8.8 Summary of actual and network test responses. The data in this table refer to normalised prototype values. Actual response refer to the result from a progressive clustering algorithm. Note: *Case C* response refers to the single output network of Fig. 8.10. The responses of cases *A* and *B* refer to the multiple outputs network of Fig. 8.11. Incorrect response from the network is shown shaded. Note: There are no prototype errors in *Case A* response.

8.3.5 Conclusions

A progressive fuzzy clustering algorithm and its neural network implementation have been reviewed. The algorithm consists of a simple structure that can easily accommodate other fuzzy clustering models. A method to perform progressive clustering using the neural network to generate the prototypes and an external program to extract the clusters from the prototype has been presented. It has been demonstrated that a fuzzy neural network predicts the locations of cluster prototypes more reliably compared to a conventional neural network. Test results indicate that errors less than 5 % are achievable.

Chapter 9

Conclusions and Future Research Directions

This chapter summarises the main conclusions of the thesis in Section 9.1 and presents suggestions for future research directions in Section 9.2. A brief description of the demonstration programs available for the thesis is given in Section 9.3.

9.1 Conclusions

9.1.1 Theory Development

The four major pattern recognition theories that influenced the development of the CPCM's fuzzy clustering models have been presented in Chapter 2. The Bayes decision theory contributes the principles for optimum classification and discriminant functions for the separation of classes. The KNN algorithm was derived from partitional clustering theory as a non-parametric approximation of the Bayes decision rule. Using a fuzzy neural network, an attempt was made to model the prototype from a progressive clustering algorithm. Of these pattern recognition theories, the major contribution to the CPCM algorithms was derived from fuzzy clustering theory, particularly from the FCM model.

In Chapter 2, two new fuzzy clustering algorithms were developed. These were called the PFCM and EPCM algorithms. In Chapter 3, a new SFM algorithm was developed. Both PFCM and EPCM are possibilistic algorithms with possibilistic memberships. PFCM extends FCM in three major ways: (i) the capability to generate more varieties of membership functions, (ii) the capability to adjust the cluster boundary and profile to con-

trol the selection of cluster points and (iii) improved clustering at local centroids via the three cluster parameters α, m and p. PFCM extends the clustering properties of FCM. EPCM improves the stability of the possibilistic function to a range of alphacuts, resulting in less sensitivity to FCM estimates of the eta factor (or intra-cluster distance) and thus improved centring of prototypes at local centroids. SFM (in Section 3.2.3.1) has a promising potential for pattern recognition applications involving either the progressive or global clustering schemes. It uses membership like a dissimilarity index, in contrast to FCM. More significantly, SFM automatically generates a single cluster via a global clustering mechanism that is identical to FCM. We are not aware of any documented fuzzy clustering algorithms with this unusual clustering feature. The extension of SFM by a similar approach to the extension of FCM by PFCM contains implications for higher clustering efficiency and more interesting clustering possibilities (like FCM without the a priori c clusters). The fundamentals of PFCM and SFM provided in the thesis should facilitate advanced structural development of these algorithms for pattern recognition applications.

The CPCM approach described in detail in Chapter 3, demonstrates a fundamentally new method of fuzzy clustering by extending the scope of conventional fuzzy clustering, which is typically global and cluster validity dependent, into the realms of progressive fuzzy clustering and non cluster validity dependent. The CPCM approach was designed to realise some of the following advantages: (i) higher data processing efficiency, up to 250 times faster than FCM in comparison tests (compare Tables 4.1 and 4.2) using real images in 256×256 resolution, (ii) an alternative method that allows independently optimised cluster parameters in the objective function and thus approximates a solution to analytically intractable problems (see Section 6.2), and (iii) automatically determines the number of clusters that agrees well with subjective interpretation (see Fig. 6.3), without the need for secondary cluster validity verification like FCM. To enable objective clustering performance comparison of the FCM, KNN and CPCM algorithms, three new cluster validity indices were developed in Section 3.3.

9.1.2 Application Development – Fuzzy Clustering Methods

Applications of CPCM fuzzy clustering methods have been presented to solve three problem areas of pattern recognition. These problem contexts are: (i) region segmentation, (ii) boundary detection and (iii) general pattern recognition.

9.1.2.1 Region Segmentation

In chapter 4, application algorithms that successfully detect wool contaminants or tile surface defects have been presented to demonstrate the superior clustering performance of CPCM compared to FCM. In clustering real images of high resolutions, the CPCM clustering algorithm clearly has a significant advantage in processing speed compared to FCM. Moreover, because of the sequential order in which clusters are removed, CPCM can detect small scale defect patterns more accurately than FCM.

9.1.2.2 Boundary Detection

Application algorithms that successfully detect boundary features characterised by noise, fragmentation and occlusion, such as linear boundaries in Chapter 5 and circular boundaries in Chapter 6, have been presented to demonstrate useful extensions of the basic FCM algorithm. This was achieved by a modification of the metric d_k to include specific cluster parameter such as the line gradient or the cluster radius. Although fuzzy clustering solutions based on the FCM model can be used, we have instead, adopted optimising methods from other models. These consist of: (i) line equation from geometry, (ii) OCF equations from statistics and (iii) CA equations from heuristics. Our procedure demonstrated the flexibility of the CPCM approach and the interesting clustering solutions from the CPCM framework. However, the context of the clustering process remains fuzzy in character, as is the CPCM framework.

The clustering method adopted for circle detection is quite different from the method for line detection, because the particular choice of equations used lacks a general character. If a general form of a quadratic equation is used, such as in [Krishnapuram et al., 1995], the clustering method to detect any of the quadratic curves will be similar.

Unlike the PCM, FKE or FKR solutions, CPCM is not as sensitive to initial cluster parameter conditions because there are numerous cluster solutions in the CPCM's solution space. Moreover, the specification of CPCM cluster parameters such as N_{min} and α_t, have meaningful notions related to the structure in data, unlike the abstract parameters (such as V_0 or U_0) in other fuzzy algorithms, which are usually selected on a random basis.

9.1.2.3 General Pattern Detection

In the application presented in Chapter 7, a combination of different pattern recognition methods involving image preprocessing, fuzzy clustering and model feature matching was successfully applied to correctly detect and identify a local object feature of any arbitrary pattern. The accurate determination of cluster centres from a ring-shape cluster detection algorithm improved the accuracy of the optimum match angle. The application also demonstrates the advantage of fuzzy clustering for locating centre compared to a conventional technique such as the centroid of area method (compare Figs. 7.7 and 7.8, from the same object). The speed enhancements in the particular implementation of the pattern matching algorithm improved the search speed by a factor of five. This was achieved using a combination of windowing and data sectoring techniques, and the similarity index.

9.1.3 Application Development – Fuzzy Neural Methods

Three fuzzy neural network configurations have been presented in Chapter 8 to demonstrate the greater response (or classification) accuracy compared to conventional neural networks. The first application in Section 8.1 presents an illumination insensitive method called the SFPM method to improve general object recognition for a range of illumination conditions. The second application in Section 8.2 improved general object recognition by matching correlation coefficients generated from a fuzzy neural network. The third application from Section 8.3 presents a fuzzy neural configuration that performs mapping of cluster prototypes from normalised gray levels of histogram data. In the third application, the output response of a conventional neural network suffered from "cross-talk" due to interference from interconnected weights. This problem was mitigated with a fuzzy network design, thus the improved classification accuracy.

9.2 Future Research Directions

This research has uncovered a number of interesting areas for future research. The following suggestions for future research are made along the three principal lines given below:

9.2.1 Extension of CPCM

CPCM has a flexible framework that can be readily extended in a number of ways, described by the following:

9.2.1.1 Cluster Density Control

The location of clusters in the CPCM framework is based on the current data centroid. This seems adequate for most clustering applications, but provides no selective control over the density of clusters found. Possibilistic clustering algorithms such as the PCM or EPCM offer scope for detecting dense clusters via the factor η which determines the cluster bandwidth (refer Section 2.4.7).

9.2.1.2 Prototype Estimation

Prototype estimation from the data centroid in CPCM is a commonly used technique (like the KNN algorithm). A weakness in this approach is that it does not discriminate differences between low and high density regions for cluster development. A low density region yields insufficient points for cluster development, resulting in the removal of these points from subsequent cluster consideration. Consequently, some loss of useful data from the clustering process is inevitable. The number of missing data, however, does not affect the clustering result to a significant extent because the minimum cluster size N_{min} provides some control over the extent of missing data. Nevertheless, this is not a desirable solution.

A better solution may be provided by SFM, derived in Section 3.2.3.1. This algorithm seems to give a good estimate of the densest cluster prototype in any clustering sequence. Since, the prototype estimate is an additional operation to the normal CPCM clustering sequence, there is more computation effort in obtaining the cluster result. To improve processing efficiency, it may be possible to use SFM to a greater degree or perhaps even in a major way, for the determination of cluster prototypes.

9.2.1.3 Exploration of Other Cluster Structures

Most of the cluster structures considered for applications were of either linear or circular varieties from both region and edge types. Adding to this list with other graphic primitives will increase the detection capability of the CPCM clustering methods. The equations to detect elliptic clusters can be obtained from either a combination of possibilistic membership with fuzzy covariance matrix (Section 2.4.6) or from the parametrized prototypes (Section 2.4.8). Clusters that are shaped as squares can be detected using either a 1-norm or a sup-norm (see [Bezdek, 1995]). One way to solve more sophisticated cluster forms is by partial aggregation of cluster substructures, such as lines and various types of arcs, ellipses and circles. The graphic elements segmented by low level algorithms can be combined to form a meaningful composite cluster structure.

9.2.2 Exploration of PFCM Clustering Possibilities

Potentially, PFCM offers greater possibilities for clustering in terms of the greater expressive power of membership functions, most of which are as yet largely unexplored. An example has been given in Section 2.4.7 to illustrate the crisp partitioning of data sets with noise points or points at boundaries of clusters. The PFCM algorithm can directly extract representative cluster points and so eliminate points at the fringe of membership. Forcing an arbitrary assignment of these points to any cluster, in the case of FCM, do not give a reasonable cluster interpretation. This is also possible with possibilistic algorithms such as PCM or EPCM, but the procedure is less direct. Possibilistic algorithms require initial cluster prototype estimates to find good clusters. These estimates are usually obtained from FCM or other clustering algorithms like the KNN.

Clustering around local centroids provides a degree of insensitivity to noise points or fuzzy points at or near the cluster boundaries. An example has been given in Section 2.4.7 to show that clustering at centroids can be obtained from either FCM or PFCM by a judicious selection of the distance metric exponent p and the fuzzifier exponent m. In practice, it is satisfactory to assume $m = 2$, but the selection of the optimal cluster parameter p involves some trial and error since no analytic forms exist. Therefore, an analytic expression or empirical formulation of p would be quite useful for robust algorithm design. A reason-

able solution seems possible by taking into consideration the inter-cluster and the intra-cluster distances.

9.2.3 Unifying Cluster Constants

The cluster constants such as the alphacut α_t, the fuzzifier exponent m, the distance metric exponent p in FCM and the α of CPCM, can be difficult to use without sufficient experience. There is clearly a need to address the needs of the end user. In this thesis, tentative attempts have been made to understand the role of these constants in clustering performance (see Sections 2.4.5 and 2.4.7 and parts of Chapters 5 and 6), but not in sufficient depth, for several reasons. Firstly, this topic involves more intensive investigation of clustering characteristics than could be adequately given within the limited scope of this thesis. Secondly, this topic is not the major focus of the thesis. A proper research program is required to develop a unifying approach to this problem.

9.3 Demonstration Programs

Demonstration programs are available, containing both source codes in C programming language and executable codes to run under DOS and MS Windows95 environments for the experiments of Chapters 3 to 7 (inclusive). These may be obtained by contacting the author or from the Head of the Department of Electrical and Electronic Engineering, Associate Professor Patrick Leung.

Included in Appendix F is a demonstration program (C source code only) for the round cluster structure with variable η algorithm discussed in Chapter 3. This program contains several features that explain the details of the following:

1. The CPCM framework.
2. The similarity coefficients used in Chapter 3.
3. The variable η algorithm.
4. The reclustering procedure for more accurate prototype location.
5. Other necessary procedures to support CPCM.

References

Adimari M., Masciangelo S., Borghesi L. and Vernazza G. (1988). A knowledge based approach to industrial scene analysis: shadows and reflexes detection, in *Image Analysis and Processing II*, Cantoni V. et al. (eds.), Plenum Press, pp. 101-110.

Anderberg, M.R. (1973). *Cluster Analysis for Applications*, Academic Press.

Backer E. and Jain A.K. (1981). A clustering performance measure based on fuzzy set decomposition, *IEEE Trans. Patter Anal. and Machine Intell.*, PAMI-3, No. 1 , pp. 66-75.

Ball G.H. and Hall D.J. (1967). A clustering technique for summarizing multivariate data, *Behav. Sci.*, Vol. 12, pp. 153-155.

Bellman R., Kalaba R. and Zadeh L. (1966). Abstraction and pattern classification, *Journal of Math. Anal. Appl.*, Vol. 13, pp. 1-7.

Bezdek J.C. (1973). *Fuzzy Mathematics in Pattern Classification*, Ph.D Thesis, Applied Math. Center, Cornell University, Ithaca.

Bezdek J.C. (1974). Cluster validity with fuzzy sets, *Journal of Cybern.*, Vol. 3, No. 3, pp. 58-72.

Bezdek J.C. (1981). *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum Press, N.Y.

Bezdek J.C. (1995a). Fuzzy and network models for clustering and classification, *IEEE Conf on Neural Networks*, Tutorial, University of Western Australia, Perth.

Bezdek J.C. (1995b). An introduction to fuzzy pattern recognition models, *3rd Australian and New Zealand Conf. on Intelligent Information Systems*, Tutorial, University of Western Australia, Perth.

Bezdek J.C., Windham M.P. and Ehrlich, R. (1980). Statistical parameters of fuzzy cluster validity functionals, *Int. Journal of Comp. and Inf. Sci.*, Vol. 9, No. 4, pp. 232-336.

Bezdek J.C., Coray C., Gunderson R. and Watson J. (1981a). Detection and characterization of cluster substructure. I. Linear structure: Fuzzy *c*-lines, *SIAM, J. of Applied Mathematics*, Vol. 40, pp. 339-357.

Bezdek J.C., Coray C., Gunderson R. and Watson J. (1981b). Detection and characterization of cluster substructure. II. Fuzzy *c*-varieties and convex combinations thereof, *SIAM, J. of Applied Mathematics*, Vol. 40, pp. 358-372.

Bezdek J.C. and Pal. S.K. (eds.), (1992). *Fuzzy models for pattern recognition: Methods that search for structures in data*, IEEE Press.

Bezdek J.C., Tsao E.C.K. and Pal N.R. (1992). Fuzzy Kohonen clustering networks, *IEEE Int. Conf. on Fuzzy Systems*, pp.1035-1043.

Bezdek J.C. and Kerr D. (1994). Training edge detecting fuzzy neural networks with model based examples, *Proc. IEEE Conf. on Fuzzy Systems*, Vol. 2, pp. 894-901.

Bezdek J.C., Hathaway R.J. and Pal N.R. (1995). Shell-prototype clustering models, *Proc. IEEE Int. Conf. on Neural Networks*, Western Australia, Perth, pp. 1617-1621.

Burden R.L. and Faires J.D. (1989). *Numerical Analysis*, 4th Edition, PWS-KENT.

Caudill M. and Butler C. (1992a). *Understanding Neural Networks: Computer Explorations, Volume 1: Basic Networks*, MIT Press.

Caudill M. and Butler C. (1992b). *Understanding Neural Networks: Computer Explorations, Volume 2: Advanced Networks*, MIT Press.

Chaudhuri B.B. and Kundu P. (1993). Optimum circular fit to weighted data in multi-dimensional space, *Pattern Recognition Letters*, Vol. 14, pp. 1-6.

Cohen H.A. (1993). One pass gray-scale image segmentation, *2nd Conf. on Digital Image Computing: Techniques and Applications*, Macquarie University, pp.572-578.

Cunningham R. (1981). Segmenting binary images, *Robotics Age*, Vol. 3, No. 4, pp. 121-135.

DARPA (1988). *DARPA Neural Network Study*, Fairfax, VA: AFCEA Press.

Dave R.N. (1989). Use of adaptive fuzzy clustering algorithm to detect lines in digital images, *SPIE Intelligent Robotics and Computer Vision VIII*, Vol. 1192 (2), pp. 600-611.

Davies D.L. and Bouldin D.W. (1979). A cluster separation measure, *IEEE Trans. Pattern Anal. and Machine Intell.*, Vol. 1, No. 2, pp. 224-227.

Davies E.R. (1990). *Machine Vision: Theory, Algorithms, Practicalities*, Academic Press.

Diday E. and Simon J.C. (1976). Clustering analysis, in *Communication and Cybernetics 10: Digital Pattern Recognition*, Fu K.S. (ed.), Springer-Verlag, pp. 47-93.

Dougherty E.R. (1992). *An Introduction to Morphological Image Processing*, SPIE Press.

Dubes R.C. (1987). How many cluster are best? An experiment, *Pattern Recognition*, Vol. 20, No. 6, 645-663.

Dubes R. and Jain A.K. (1980). Clustering methodologies in exploratory data analysis, in *Advances in Computers*, Yovits M.C. (ed.), Vol. 19, Academic Press, pp. 113-215.

Duda R.O. and Hart P.E. (1973). Pattern Classification and Scene Analysis, John Wiley & Sons.

Dunn J.C. (1973). A fuzzy relative of the ISODATA process and its use in detecting compact, well-separated clusters, *J. of Cybernetics*, Vol. 3, pp. 32-57.

Dunn J.C. (1974). Well separated clusters and optimal fuzzy partitions, *Journal of Cybern.*, Vol. 4-1, pp. 95-104.

Dunn J.C. (1976). Indices of partition fuzziness and the detection of clusters in large data sets, in *Fuzzy Automata and Decision Processes*, Gupta M. (ed.) Elsevier,

Everitt B.S. (1974). *Cluster Analysis*, John Wiley and Sons.

Flusser J. (1995). Object matching by means of matching likelihood coefficients, *Pattern Recognition Letters*, Vol. 16, pp. 893-900.

Forgy E. (1965). Cluster analysis of multivariate data: efficiency versus interpretability of classifications (abstract), *Biometrics*, Vol. 21, p. 768.

Freeman J.A. (1994). *Simulating Neural Networks with Mathematica*, Addison-Wesley.

Freeman J.A. and Skapura D.M. (1992). *Neural Networks: Algorithms, Applications and Programming Techniques*, Addison-Wesley.

Fu K.S. and Mui J.K. (1981). A survey on image segmentation, *Pattern Recognition*, Vol. 13, pp. 3-16.

Fukunaga K. (1972). *Introduction to Statistical Pattern Recognition*, Academic Press.

Fukushima K., Miyake S. and Takayuki I. (1983). Neocognitron: A neural network model for a mechanism of visual pattern recognition, *IEEE Trans. Systems, Man and Cybernetics*, SMC-13(5), pp. 826-834.

Gath I. and Geva A. (1989). Unsupervised optimal fuzzy clustering, *IEEE Trans. Pattern Anal. and Machine Intell.*, Vol. 11, No. 7, pp. 773-781.

Gath I. and Hoory D. (1995). Fuzzy clustering of elliptic ring-shaped clusters, *Pattern Recognition Letters*, Vol. 16, pp. 727-741.

Gitman I. and Levine M. (1970). An algorithm for detecting unimodal fuzzy sets and its application as a clustering technique, *IEEE Trans. Computers*, Vol. C-19, pp. 917-923.

Gonzalez R.C. and Woods R.E. (1992). *Digital Image Processing*, Addison-Wesley.

Gordon A.D. (1981). *Classification: Methods for Exploratory Analysis of Multivariate Data*, Chapman and Hall.

Gosh A., Pal N.R. and Pal S.K. (1993). Self-organization for object extraction using a multilayer neural network and fuzziness measure," *IEEE Trans. Fuzzy Systems*, Vol. 1, No. 1, pp.54-68.

Gowder K.C. and Krishna G. (1978). Agglomerative clustering using the concept of mutual nearest neighbourhood, *Pattern Recognition*, Vol. 10, pp. 105-112.

Grimson W.E.L. (1990). *Object Recognition by Computer: The Role of Geometric Constraints*, MIT Press.

Grossberg S. (ed.) (1988). *The Adaptive Brain*, Volume 1 and 2, North Holland Press (Elsevier).

Grossberg S. (ed.) (1989). *Neural Networks and Natural Intelligence*, The MIT Press.

Gustafson D.E. and Kessel W. (1979). Fuzzy clustering with a fuzzy covariance matrix, *Proc. IEEE, CDC*, pp. 761-766.

Hall L.O., Bensaid A.M., Clarke L.P., Velthuizen R.P., Silbiger M.S. and Bezdek J.C. (1992). A comparison of neural network and fuzzy clustering techniques in segmenting magnetic resonance images of the brain, *IEEE Trans. Neural Networks*, Vol.3 No. 5, pp. 672-682.

Hanson S.J. and Pratt L.Y. (1989). Comparing biases for minimal network construction with back propagation, in *Advances in Neural Information Processing Systems*, Touretzky D.S. (ed.), Vol. 1, Morgan Kaufmann, pp. 177-185.

Haralick R.M. and Shapiro L.G. (1992). *Computer and Robot Vision* – Volume 1, Addison-Wesley.

Haralick R.M. and Shapiro L.G. (1993). *Computer and Robot Vision* – Volume 2, Addison-Wesley.

Har-even M. and Brailovsky V.L. (1995). Probabilistic validation approach for clustering, *Pattern Recognition Letters*, Vol. 16, pp. 1189-1196.

Hartigan J.A. (1975). *Clustering Algorithms*, Wiley.

Hebb D.O. (1949). *The Organisation of Behaviour: A Neuropsychological Theory*, John Wiley & Sons.

Hertz J., Krogh A. and Palmer R.G. (1991). *Introduction to the Theory of Neural Computation*, Addison-Wesley.

Hinton G.E. (1987). Connectionist learning procedures, *Technical Report CMU-CS-87-115*, (version 2), Carnegie-Mellon University.

Hopfield J.J. (1982). Neural networks and physical systems with emergent collective computational abilities, *Proc. of the National Academy of Science*, USA, Vol. 79, pp. 2554-2558.

Hopfield J.J. and Tank D.W. (1985a). Computing with neural circuits: A model, *Science*, Vol. 233, pp. 625-633.

Hopfield J.J. and Tank D.W. (1985b). Neural computation of decisions in optimisation problems, *Biological Cybernetics*, Vol. 52, pp. 141-152, Springer-Verlag.

Hubert L.J. and Arabie P. (1985). Comparing partitions, *Journal of Classification*, Vol. 2, pp. 193-218.

Huntsberger T. and Ajjimarangsee P. (1989). Parallel self-organizing feature maps for unsupervised pattern recognition, *Int. Journal of General Systems*, Vol. 16, pp. 357-372.

Hush D.R. and Horn B. (1993). Progress in supervised neural networks: What's new since Lippmann?, *IEEE Signal Processing magazine*, pp. 8-39.

Im P.T. (1992). The pattern recognition of industrial parts using deterministic decision rule based strategy, M. Eng. Thesis, Victoria University of Technology, Victoria.

Jain A.K. (1987). Advances in statistical pattern recognition, in *Pattern Recognition Theory and Applications*, Devijver P.A. and Kittler J. (eds), Springer-Verlag, pp. 1-19.

Jain A.K. and Dubes R.C. (1988). *Algorithm for Clustering Data*, Prentice-Hall.

Jain R., Kasturi R. and Schunck B.G. (1995). *Machine Vision*, McGraw-Hill.

James M. (1985). *Classification Algorithms*, John Wiley & Sons.

Jarvis R.A. (1978). Shared near neighbour maximal spanning trees for cluster analysis, *Proc. of the 4th Int. Joint Conf. on Pattrn Recognition*, pp. 308-313.

Jensen R.E. (1969). A dynamic programming algorithm for cluster analysis, *Operations Research*, Vol. 17, pp. 1034-1057.

Jolion J. M. (1994). Computer vision methodologies, *Computer Vision Graphics Image Processing*, Vol. 59, No. 1, pp. 53-71.

Juan A. and Vidal E. (1994). Fast *K*-means like clustering in metric space, *Pattern Recognition Letters*, Vol. 15, pp. 19-25.

Kandel A. (1982). *Fuzzy techniques in pattern recognition*, John Wiley & Sons.

Kaufman L. and Rousseeuw P.J. (1990). *Finding Groups in Data*, John Wiley & Sons.

Kittler J. (1976). A locally sensitive method for cluster analysis, *Pattern Recognition*, Vol. 8, pp. 22-33.

Kohonen T. (1988). An introduction to neural computing, *Neural Networks*, Vol. 1, No. 1, pp. 3-16.

Koontz W.L.G., Narendra P.M. and Fukunaga K. (1975). A branch and bound clustering algorithm, *IEEE Trans. on Computers*, Vol. 23, pp.908-914.

Kosko, B. (1992). *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*, Prentice-Hall.

Kosko, B. (1993). *Fuzzy Thinking: The New Science of Fuzzy Logic*, Hyperion.

Krishnapuram R. and Gupta S. (1992). Edge Detection in Range Images through Morphological Residue Analysis, *Proc. IEEE Computer Soc. Conf. on Computer Vision and Pattern Recognition*, pp. 630-632.

Krishnapuram R. and Keller J.M. (1992). Fuzzy set theoretic approach to computer vision, *IEEE Int. Conf. on Fuzzy Systems*, pp.135-142.

Krishnapuram R. and Keller J.M. (1993a). A possibilistic approach to clustering, *IEEE Trans. Fuzzy Systems*, Vol. 1, No. 2, pp. 98-110.

Krishnapuram R. and Keller J.M. (1993b). Quantitative analysis of properties and spatial relations of fuzzy image regions, *IEEE Trans. Fuzzy Systems*, Vol. 1, No. 3, pp. 222-233.

Krishnapuram R. (1994). Generation of membership functions via possibilistic clustering, *Proc. of the 3rd IEEE Conf. on Fuzzy Systems*, pp. 902-908.

Krishnapuram R., Frigui H. and Nasraoui O. (1995). Fuzzy and possibilistic shell clustering algorithms and their application to boundary detection and surface approximation – Part 1, *IEEE Trans. Fuzzy Systems*, Vol. 3, No. 1, pp. 29-43.

Lawrence K.D. and Arthur J.L. (1990). *Robust Regression: Analysis and Applications*, Marcel Dekker, Inc.

Leavers V.F. (1992). *Shape Detection Using the Hough Transform*, Springer-Verlag.

Lee S. and Kil R.M. (1991). A gaussian potential function network with hierarchically self-organising learning, *Neural Networks*, Vol. 4, pp. 207-224.

Lefkovitch L.P. (1980). Conditional clustering, *Biometrics*, Vol. 36, pp. 43-58.

Lippmann R.P. (1987). An introduction to computing with neural networks, *IEEE Acoustics, Speech and Signal Processing Magazine*, Vol. 4, No. 2, pp. 4-22.

Lorentz G.G. (1976). The 13th problem of Hilbert, in *Mathematical Developments Arising from Hilbert Problems*, Browder F.E. (ed.), American Mathematical Society, Providence, R.I.

Man Y. and Gath I. (1994). Detection and separation of ring-shaped clusters using fuzzy clustering, *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. 16, No. 8, pp. 855-861.

Masters T. (1993). *Practical Neural Network Recipes in C++*, Academic Press.

McCulloch W.S. and Pitts W.H. (1943). A logical calculus of the ideas imminent in nervous activity, *Bulletine of Mathematical Biophysics*, Vol. 5, pp. 115-133.

McQueen J.B. (1967). Some methods of classification and analysis of multivariate observations, *Proc. of Fifth Berkeley Symposium on Math. Statistics and Probability*, pp. 281-297.

Minsky M. And Papert S. (1969). *Perceptrons: An Introduction to Computational Geometry*, MIT Press.

Mokhtarian F. (1995). Silhouette-based isolated object recognition through curvature scale space, *IEEE Trans. Pattern Anal. and Machine Intell.*, Vol. 17, No. 5, pp. 539-544.

Moreau J.V. and Jain A.K. (1987). The bootstrap approach to clustering, in *Pattern Recognition Theory and Applications*, Devijver P.A. and Kittler J. (eds), Springer-Verlag, pp. 63-71.

Moody J. And Darken C.J. (1989). Fast learning in networks of locally-tuned processing units, *Neural Computation*, Vol. 1, pp. 281-293.

Musavi M.T., Ahmed W., Chan K.H., Faris K.B. and Hummels D.M. (1992). On the training of radial basis function classifiers, *Neural Networks*, Vol. 5, pp. 595-603.

Niemann H., Sagerer G.F., Schröder S. and Kummert F. (1990). ERNEST: A semantic network for pattern understanding, *IEEE Trans. Pattern Anal. and Machine Intell.*, Vol. 12, No. 9, pp. 883-905.

Pal S.K. (1992). Fuzzy sets in image processing and recognition, *IEEE Int. Conf. on Fuzzy Systems*, San Diego, California, USA, pp. 119-126.

Pal S.K. and King R.A. (1983). On edge detection of X-Ray images using fuzzy sets, *IEEE Trans. Pattern Anal. and Machine Intell.*, Vol. PAMI-5, No. 1, pp. 69-77.

Pal S.K. and Majumder D.K.D. (1986). *Fuzzy mathematical approach to pattern recognition*, John Wiley & Sons.

Pao Y.H. (1989). *Adaptive Pattern Recognition and Neural Networks*, Addison-Wesley.

Parker J.R. (1994). *Practical Computer Vision Using C*, John Wiley & Sons.

Postaire J.G., Zhang R.D. and Lecocq-Botte C. (1993). Cluster analysis by binary morphology, *IEEE Trans. Pattern Anal. and Machine Intell.*, Vol. 15, pp. 170-180.

Powell M.J.D. (1985). Radial basis functions for multivariate interpolation: A review, *Technical Report DAMPT 1985/NA12*, Dept. of app. math. and theor. physics, Cambridge University, Cambridge.

Rao M.R. (1971). Cluster analysis and mathematical programming, *Journal of the American Statistical Association*,Vol. 66, pp. 622-626.

Rao V.B. and Rao H.V. (1995). *C++ Neural Networks and Fuzzy Logic*, MIS Press.

Rhee F.C.H. and Krishnapuram R. (1994). Generation of fuzzy rules involving spatial relations for computer vision, *Proc. IEEE Conf. on Fuzzy Systems*, pp. 2014-2019.

Rosenblatt F. (1959). Two theorems of statistical separability in the perceptron, in *Mechanisation of Thought Processes: Proc. of Symposium No. 10*, held at the National Physical Laboratory, November 1958, H.M. Stationary Office, London, Vol. 1, pp. 421-456.

Rosenblatt F. (1962). *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*, Spartan, Washington, D.C.

Ruck D.W., Rogers S.K., Kabrisky M., Oxley M.E. and Suter B.W. (1990). The multilayer perceptron as an approximation to a Bayes optimal discriminant function, *IEEE Trans. Neural Networks*, Vol. 1, No. 4, pp. 296-298.

Rumelhart D.E., Hinton G.E. and Williams R.J. (1986). Learning internal representations by error propagation, in *Parallel and Distributed Processing: Explorations in the Microstructures of Cognition, Vol. 1, Foundations*, Rumelhart D.E. and McClelland J.L. (eds.), MIT Press, pp. 318-362.

Ruspini E.H. (1969). A new approach to clustering, *Information and Control*, Vol. 15, pp. 22-32.

Ruspini E.H. (1970). Numerical methods for fuzzy clustering, *Inf. Sci.*, Vol. 2, pp. 319-350.

Serra J. (ed.) (1988). *Image Analysis and Mathematical Morphology*, Vol. 2, Academic Press.

Shannon C.E. (1948). A mathematical theory of communication, *Bell Syst. Tech. Journal*, Vol. XXVII-3, pp. 379-423.

Simpson P.K. (1990). *Artificial Neural Systems: Foundations,Paradigms, Applications and Implementations*, Pergamon Press.

Sonka M., Hlavac V. and Boyle R. (1994). *Image Processing, Analysis and Machine Vision*, Chapman & Hall Computing.

Sprecht D.F. (1990a). Probabilistic neural networks, *Neural Networks*, Vol. 3, No. 1, pp. 109-118.

Sprecht D.F. (1990b). Probabilistic neural networks and the polynomial adaline as complementary techniques for classification, *IEEE Trans. Neural Networks*, Vol. 1, No. 1, pp. 111-121.

Sum J. and L.W. Chan (1994). Alternative membership function for sequential fuzzy clustering, *IEEE Int. Conf. on Fuzzy Systems*, pp. 1846-1851.

Tou J.T. and Gonzalez R.C. (1974). *Pattern Recognition Principles*, Addison-Wesley.

Ventura J.A., Nain L. Y. and Wan W. (1995). Optimal matching of general polygons based on the minimum zone error, *Pattern Recognition Letters*, Vol. 16, pp. 1125-1136.

Welstead S.T. (1994). *Neural Network and Fuzzy Logic Applications in C/C++*, John Wiley & Sons.

Weigend A.S., Rumelhart D.E. and Huberman B.A. (1990). Back-propagation, weight elimination and time series prediction, in *Proc. of the 1990 Connectionists Models Summer School*, Morgan Kaufmann, pp. 65-80.

Weigend A.S., Huberman B.A. and Rumelhart D.E. (1991). Predicting sunspots and exchange rates with connectionists networks, in Casdagli M. and Eubank S. (eds.), *Nonlinear Modeling and Forecasting, SFI Studies in the Sciences of Complexity*, Vol. 12, Addison-Wesley.

Xie X.L. and Beni G.A. (1991). A validity measure for fuzzy clustering, *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. 3, No. 3, pp. 841-846.

Yager R.R. and Zadeh L.A. (eds.) (1992). *An Introduction to Fuzzy Logic Applications in Intelligent Systems*, Kluwer Academic Press.

Yang J. and Li X. (1995). Boundary detection using mathematical morphology, *Pattern Recognition Letters*, Vol. 16, pp. 1277-1286.

Zadeh, L.A. (1965). Fuzzy sets, *Inf. Control.*, Vol. 8, pp. 338-353.

Zahn C.T. (1971). Graph-theoretical methods for detecting and describing Gestalt clusters, *IEEE Trans. Computers*, Vol. 20, pp. 68-86.

Zurada J.M. (1992). *Artificial Neural Systems*, West Publishing Co.

Appendix A. Vectors in Real *n*-Space

The material in this appendix is intended to define the usage of vector definitions and properties implicitly assumed in the thesis. In pattern recognition work, it is useful to define an n-space (n-dimensional coordinates) for each feature vector or point. This space is alternatively known as the Euclidean n-space. Axioms for the inner product space are defined for the Euclidean n-space. The inner product introduces the concept of length and distance in the inner product space.

Definition A.1: If n is a positive integer, then an ordered-n-tuple is a sequence of n real numbers $(a_1,a_2,\ldots,a_n)$. The set of all ordered n-tuples is called n-space and denoted by $\Re^n$.

In n-space, the symbol $(a_1,a_2,\ldots,a_n)$ can be interpreted as a point, in which case $a_1,a_2,\ldots,a_n$ are the coordinates of a point or it can be interpreted as a vector, in which case $a_1,a_2,\ldots,a_n$ are the components of a vector. A vector is alternatively denoted as

$$\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ . \\ . \\ a_n \end{bmatrix} \text{ or } \mathbf{a} = [a_1,a_2,\ldots,a_n]^T.$$

A zero vector in $\Re^n$ is $\mathbf{0} = [0,0,\ldots,0]^T$. A negative vector (or additive inverse) $\mathbf{a}$ is denoted by $-\mathbf{a} = [-a_1,-a_2,\ldots,-a_n]^T$.

The following are some basic definitions of vector operations.

Definition A.2a: Two vectors $\mathbf{a} = [a_1,a_2,\ldots,a_n]^T$ and $\mathbf{b} = [b_1,b_2,\ldots,b_n]^T$ are equal if

$$a_1 = b_1,\ a_2 = b_2,\ldots,\ a_n = b_n.$$

Definition A.2b: The sum of two vectors $\mathbf{a} = [a_1,a_2,\ldots,a_n]^T$ and $\mathbf{b} = [b_1,b_2,\ldots,b_n]^T$ is defined as $\mathbf{a} + \mathbf{b} = [a_1 + b_1, a_2 + b_2,\cdots,a_n + b_n]^T$.

Definition A.2c: If k is any scalar, the scalar multiple $k\mathbf{a}$ is defined by

$$k\mathbf{a} = [ka_1,ka_2,\ldots,ka_n]^T$$

Theorem A.1 (addition and scalar multiplication): If $\mathbf{a} = [a_1,a_2,\ldots,a_n]^T$, $\mathbf{b} = [b_1,b_2,\ldots,b_n]^T$ and $\mathbf{c} = [c_1,c_2,\ldots,c_n]^T$ are vectors in $\Re^n$ and k and l scalars, then:

(a) $\mathbf{a} + \mathbf{b} = \mathbf{b} + \mathbf{a}$

(b) $\mathbf{a} + (\mathbf{b} + \mathbf{c}) = (\mathbf{a} + \mathbf{b}) + \mathbf{c}$

(c) $\mathbf{a} + \mathbf{0} = \mathbf{0} + \mathbf{a} = \mathbf{a}$

(d) $\mathbf{a} + (-\mathbf{a}) = \mathbf{a} - \mathbf{a} = 0$

(e) $k(l\mathbf{a}) = (kl)\mathbf{a}$

(f) $k(\mathbf{a} + \mathbf{b}) = k\mathbf{a} + k\mathbf{b}$

(g) $(k+l)\mathbf{a} = k\mathbf{a} + l\mathbf{b}$

(h) $1\mathbf{a} = \mathbf{a}$

Proofs of Theorem A.1(a) to E.1(h) may be found in a textbook on linear algebra.

Definition A.3: If $\mathbf{a} = [a_1,a_2,\ldots,a_n]^T$ and $\mathbf{b} = [b_1,b_2,\ldots,b_n]^T$ are vectors in $\Re^n$, then the Euclidean inner (dot or scalar) product $\mathbf{a} \cdot \mathbf{b}$ is defined by

$$\mathbf{a} \cdot \mathbf{b} = a_1b_1 + a_2b_2 + \cdots + a_nb_n = \sum_{k=1}^{n} a_k b_k$$

Theorem A.2 (Euclidean inner product theorem): If $\mathbf{a} = [a_1,a_2,\ldots,a_n]^T$, $\mathbf{b} = [b_1,b_2,\ldots,b_n]^T$ and $\mathbf{c} = [c_1,c_2,\ldots,c_n]^T$ are vectors in $\Re^n$ and k is any scalar, then

(a) $\mathbf{a} \cdot \mathbf{b} = \mathbf{b} \cdot \mathbf{a}$

(b) $(\mathbf{a} + \mathbf{b}) \cdot \mathbf{c} = \mathbf{a} \cdot \mathbf{b} + \mathbf{b} \cdot \mathbf{c}$

(c) $(k\mathbf{a}) \cdot \mathbf{b} = k(\mathbf{a} \cdot \mathbf{b})$

(d) $\mathbf{a} \cdot \mathbf{a} \geq 0$; and $\mathbf{a} \cdot \mathbf{a} = 0$ if and only if $\mathbf{a} = \mathbf{0}$

Proofs of Theorem A.2(a) to (d) may be found in a textbook on linear algebra.

Definition A.4: The Euclidean norm or length of a vector $\mathbf{a} = [a_1,a_2,\ldots,a_n]^T$ in $\Re^n$ is defined by $\|\mathbf{a}\|_2 = \|\mathbf{a}\| = \sqrt{(\mathbf{a} \cdot \mathbf{a})} = \sqrt{a_1^2 + a_2^2 + \cdots + a_n^2} = \left(\sum_{k=1}^{n} a_k^2 \right)^{1/2}$

Definition A.5: The Euclidean distance between a vector $\mathbf{a} = [a_1,a_2,\ldots,a_n]^T$ in $\Re^n$ and $\mathbf{b} = [b_1,b_2,\ldots,b_n]^T$ in $\Re^n$ is defined by

$$d(\mathbf{a},\mathbf{b}) = \|\mathbf{a} - \mathbf{b}\| = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \cdots + (a_n - b_n)^2} = \left(\sum_{k=1}^{n} (a_k - b_k)^2 \right)^{1/2}$$

In fuzzy clustering, the Euclidean distance (or norm or metric) $d(\mathbf{a},\mathbf{b})$ is alternatively denoted as d_{ik} to represent the distance between the ith cluster prototype vector $\mathbf{v}_i$ and the kth feature (or point) vector (or data) $\mathbf{x}_k$. Since matrix operations are useful to fuzzy cluster analysis, the inner product of vectors **a** and **b** is alternatively represented by a matrix formula

$$\mathbf{a} \cdot \mathbf{b} = \mathbf{a}^T\mathbf{b}$$

The preceding discussion presents the Euclidean inner product on the Euclidean n-space $\Re^n$ of which Theorem A.2 is its most important properties. These properties may be defined for a general real vector space V as follows:

Definition A.6: An inner product on a real vector space V is a function that associates a real number $\langle \mathbf{a}, \mathbf{b} \rangle$ with each pair of vectors **a** and **b** in V in such a way that the following axioms are satisfied for all vectors **a**, **b** and **c** in V and all scalars k.

(a) $\langle \mathbf{a}, \mathbf{b} \rangle = \langle \mathbf{b}, \mathbf{a} \rangle$ (symmetry axiom)

(b) $\langle \mathbf{a} + \mathbf{b}, \mathbf{c} \rangle = \langle \mathbf{a}, \mathbf{c} \rangle + \langle \mathbf{b}, \mathbf{c} \rangle$ (additivity axiom)

(c) $\langle k\mathbf{a}, \mathbf{b} \rangle = k\langle \mathbf{a}, \mathbf{b} \rangle$ (homogeneity axiom)

(d) $\langle \mathbf{a}, \mathbf{a} \rangle \geq 0$; and $\langle \mathbf{a}, \mathbf{a} \rangle = 0$ if and only if $\mathbf{a} = \mathbf{0}$ (positivity axiom)

The above four axioms define a real inner product space. The Euclidean inner product may be modified by weighting each term, such as in the Mahalanobis norm.

Definition A.7: Let $\mathbf{a} = [a_1, a_2, \ldots, a_n]^T$ be a vector in $\Re^n$. Let W and A be invertible symmetric $n \times n$ matrices. If $\mathbf{a} \cdot \mathbf{a}$ is the Euclidean inner product on $\Re^n$ then

$$\langle \mathbf{a}, \mathbf{a} \rangle = \mathbf{a}^T A^T A \mathbf{a} = \mathbf{a}^T W \mathbf{a} = \langle \mathbf{a}, \mathbf{a} \rangle_W = \|\mathbf{a}\|_W^2$$

define a weighted Euclidean inner product norm on $\Re^n$ called the inner product generated by W. This norm is also called a variable W norm. If the positive weights of W are diagonal, then $\langle \mathbf{a}, \mathbf{a} \rangle = \sum_{k=1}^{n} w_k a_k^2$. Likewise, a weighted inner product distance may be defined by $\langle \mathbf{a} - \mathbf{b}, \mathbf{a} - \mathbf{b} \rangle_W = \|\mathbf{a} - \mathbf{b}\|_W^2$ and $\langle \mathbf{a}, \mathbf{b} \rangle = \mathbf{a}^T A^T A \mathbf{b} = \mathbf{a}^T W \mathbf{b} = \langle \mathbf{a}, \mathbf{b} \rangle_W$.

Appendix B. Proof of Theorem 2.4.1 (Fuzzy *c*-Means)

The Fuzzy *c*-Means algorithm or FCM [Bezdek, 1981] is an important development in fuzzy clustering history. It is virtually the basis of most modern versions of fuzzy clustering algorithms, some of which are presented in Section 2.4 [Man and Gath, 1994; Gath and Hoory, 1995]. Therefore, it is important to understanding the basic structure of FCM to understand the character of modern fuzzy clustering methods. However, there is another cogent reason to do so. The structure of FCM admits a *further* generalisation, on closer scrutiny. Evidence of this is demonstrated by the Possibilistic Fuzzy *c*-Means (PFCM) algorithm, described in Section 2.4.5 and derived in Appendix C.

For the notations, distance norm, data set and fuzzy partitions of Section 2.4.4, let the membership of FCM satisfies the three conditions

$$u_{ik} \in [0,1] \quad \forall i,k, \quad 0 < \sum_{k=1}^{N} u_{ik} < N \quad \forall i \quad \text{and} \quad \sum_{i=1}^{c} u_{ik} = 1 \quad \forall k \tag{B.1}$$

Using the method of Lagrange multipliers, the objective function $J_{m,p}$ in the expression

$$J_{m,p}(U,V) = \sum_{k=1}^{N}\sum_{i=1}^{c} u_{ik}^{m} d_{ik}^{p} \qquad (\text{for all } m > 1, p > 0) \tag{B.2}$$

may be minimised with respect to u_{ik} to give (with a change of subscripts)

$$m u_{st}^{m-1} d_{st}^{2} = \lambda$$

for $p = 2$ (as in [Bezdek, 1981]) and may be simplified to

$$u_{st} = \left(\frac{\lambda}{m}\right)^{\frac{1}{m-1}} \left(\frac{1}{d_{st}^{2}}\right)^{\frac{1}{m-1}} \tag{B.3}$$

Applying the membership constraint of (B.1) to (B.3), with a change of subscript gives

$$\sum_{j}^{c} u_{jt} = \left(\frac{\lambda}{m}\right)^{\frac{1}{m-1}} \sum_{j=1}^{c} \left[\frac{1}{d_{jt}^{2}}\right]^{\frac{1}{m-1}} = 1 \tag{B.4}$$

which results in

$$\left(\frac{\lambda}{m}\right)^{\frac{1}{m-1}} = \frac{1}{\sum_{j=1}^{c}\left[\frac{1}{d_{jt}^2}\right]^{\frac{1}{m-1}}} \tag{B.5}$$

Putting (B.5) into (B.3) gives

$$u_{st} = \frac{1}{\sum_{j=1}^{c}\left[\frac{d_{st}^2}{d_{jt}^2}\right]^{\frac{1}{m-1}}}$$

which on rearranging the subscripts, yields the standard form

$$u_{ik} = \frac{1}{\sum_{j=1}^{c}\left[\frac{d_{ik}^2}{d_{jk}^2}\right]^{\frac{1}{m-1}}} \tag{B.6}$$

J_m may be minimised for $\mathbf{v}_i$ by differentiating $J_{m,p}$ in (B.2) with respect to $\mathbf{v}_i$ and setting the resulting function to zero. The distance measure d_{ik} may be generalised with a norm weighted by a $d \times d$ positive definite matrix A_i. The A_i norm is defined as

$$d_{ik} = \|\mathbf{x}_k - \mathbf{v}_i\|_{A_i} = \sqrt{(\mathbf{x}_k - \mathbf{v}_i)^T A_i (\mathbf{x}_k - \mathbf{v}_i)} \tag{B.7}$$

which represents the distance of feature vector $\mathbf{x}_k$ from the cluster prototype $\mathbf{v}_i$. Minimising J_m with respect to $\mathbf{v}_i$ yields

$$-2\sum_{k=1}^{N} u_{ik}^m A_i \|\mathbf{x}_k - \mathbf{v}_i\| = 0$$

Since A_i is a constant under the summation k, the prototype solution is

$$\mathbf{v}_i = \frac{\sum_{k=1}^{N} u_{ik}^m \mathbf{x}_k}{\sum_{k=1}^{N} u_{ik}^m} \qquad \forall i \tag{B.8}$$

Appendix C. Proof of Theorem 2.4.2 (Possibilistic Fuzzy c-Means)

The Possibilistic Fuzzy c-Means (PFCM) algorithm is a very recent generalisation of the famous FCM algorithm, introduced in 1973. For over 23 years, major extensions to FCM focused on adapting FCM or its variants to new types of cluster substructures such as shells and hyperquadrics. Curiously, the possibility of further generalisation of FCM seems to have escaped the notice of researchers. This may reflect a predisposition to extend application development, partly because the method is new or perhaps because of a perception of limited scope for structural development. It is hoped the development of PFCM will lead to more useful insights into fuzzy clustering properties (when the implications become more transparent), reviewed in Sections 2.4.5 and 2.4.7, and promote further structural development of PFCM, alluded to in Chapter 9.

For the notations, distance measure, data set and fuzzy partitions of Section 2.4.4, let the general objective function be defined by

$$J_{m,p,\alpha}(U,V) = \sum_{k=1}^{N}\sum_{i=1}^{c} u_{ik}^{m} d_{ik}^{p} \qquad \text{(for all } m>0, p>0) \tag{C.1}$$

Let the memberships of the objective function (C.1) satisfy the following three conditions

$$u_{ik} \in [0,1] \quad \forall i,k, \quad 0 < \sum_{k=1}^{N} u_{ik} \le N \quad \forall i \quad \text{and} \quad \sum_{i=1}^{c} u_{ik}^{\alpha} = 1 \quad \forall k,\alpha \tag{C.2}$$

where α is a real valued exponent on the membership. Using the method of Lagrange multipliers, the resultant expression given by

$$\sum_{k=1}^{N}\sum_{i=1}^{c} u_{ik}^{m} d_{ik}^{p} = \lambda\left(\sum_{i=1}^{c} u_{ik}^{\alpha} - 1\right) \tag{C.3}$$

may be minimised with respect to u_{ik} to give

$$m u_{ik}^{m-1} d_{ik}^{p} = \lambda\alpha u_{ik}^{\alpha-1}$$

which after a change of subscript, simplifies to

$$u_{st} = \left(\frac{\alpha\lambda}{m}\right)^{\frac{1}{m-\alpha}} \left(\frac{1}{d_{st}^{p}}\right)^{\frac{1}{m-\alpha}} \tag{C.4}$$

From (C.4), after redefining subscripts

$$u_{jt}^{\alpha} = \left(\frac{\alpha\lambda}{m}\right)^{\frac{\alpha}{m-\alpha}} \left(\frac{1}{d_{jt}^{p}}\right)^{\frac{\alpha}{m-\alpha}}$$

and applying membership condition (C.2) to (C.4)

$$\sum_{j=1}^{c} u_{jt}^{\alpha} = \left(\frac{\alpha\lambda}{m}\right)^{\frac{\alpha}{m-\alpha}} \sum_{j=1}^{c} \left(\frac{1}{d_{jt}^{p}}\right)^{\frac{\alpha}{m-\alpha}} = 1 \tag{C.5}$$

Simplifying (C.5) gives

$$\left(\frac{\alpha\lambda}{m}\right)^{\frac{\alpha}{m-\alpha}} = \frac{1}{\sum_{j=1}^{c} \left[\frac{1}{d_{jt}^{p}}\right]^{\frac{\alpha}{m-\alpha}}}$$

or

$$\left(\frac{\alpha\lambda}{m}\right)^{\frac{1}{m-\alpha}} = \frac{1}{\left[\sum_{j=1}^{N} \left(\frac{1}{d_{jt}^{p}}\right)^{\frac{\alpha}{m-\alpha}}\right]^{\frac{1}{\alpha}}} \tag{C.6}$$

Putting (C.6) into (C.4) gives

$$u_{st} = \frac{1}{\left[\sum_{j=1}^{c} \left(\frac{d_{st}^{p}}{d_{jt}^{p}}\right)^{\frac{\alpha}{m-\alpha}}\right]^{\frac{1}{\alpha}}}$$

which on rearranging subscripts, yields

$$u_{ik} = \frac{1}{\left[\sum_{j=1}^{c} \left(\frac{d_{ik}^{p}}{d_{jk}^{p}}\right)^{\frac{\alpha}{m-\alpha}}\right]^{\frac{1}{\alpha}}} \tag{C.7}$$

At this point, it is necessary to avoid the singleton at $\mathbf{x}_k$ when $d_{ik} = 0$, by defining the membership to preclude such a point with the set $\boldsymbol{I}_k$, of (2.4.17). Using the identity $\nabla_{\mathbf{x}}(\mathbf{x}^T A\mathbf{x}) = 2A\mathbf{x}$, we may minimise the objective function (C.1) with respect to $\mathbf{v}_i$, to give

$$\sum_{k=1}^{N} u_{ik}^m d_{ik}^{p-2}(\mathbf{x}_k - \mathbf{v}_i) = \mathbf{0}$$

in which the constant matrix A can be factored out to yield the prototype solution

$$\mathbf{v}_i = \frac{\sum_{k=1}^{N} u_{ik}^m d_{ik}^{p-2}\mathbf{x}_k}{\sum_{k=1}^{N} u_{ik}^m d_{ik}^{p-2}}, \quad 1 \le i \le c \tag{C.8}$$

Appendix D. Derivation of the Delta Learning Rule for the Backpropagation Network

Interconnected weights in a neural network are like the memories of their biological counterpart. The magnitude and distribution of the weights determine the response characteristic to a given type of input signals. Because of the size of the complex network of weighted connections, weight adjustment is not a trivial task. For this reason, a learning rule or a systematic procedure by which a neural network can automatically adjust weights in a way that will optimally match the actual response to the target goals, is an important component in the architecture of a neural network. One common technique is to minimise the error between output and target, using "negative gradient descent" from differential calculus principles. The delta learning rule uses this procedure.

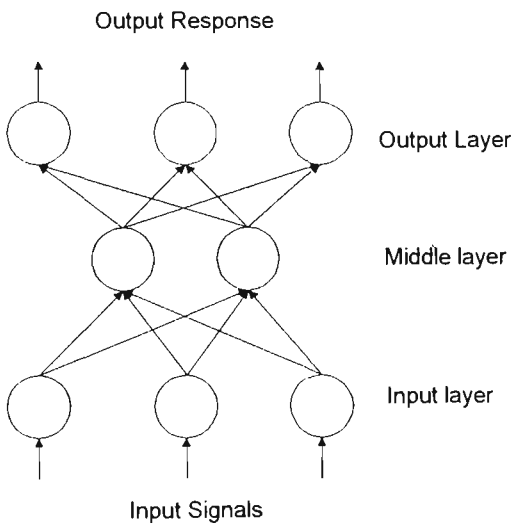


Figure D.1 Architecture of a neural network.

Referring to Fig. D.1, let the subscripts i, j and k denote the input, middle and output layers, respectively. Let $(j+1)$ denotes the layer after the jth layer and $(j-1)$ the layer before the jth layer. For a fully connected multi-layer perceptron the input to each jth layer processing unit is

$$I_{j,r} = \sum_{q=1}^{N_{j-1}} w_{(j,r),(j-1,q)} O_{j-1,q} \tag{D.1a}$$

where N_{j-1} is the number of units in the $(j-1)$ layer; O_{j-1} is the outputs from the processing units of the $(j-1)$ layer; the weight modifying the connection from the $(j-1)$ layer processing unit to the jth layer processing unit is represented by $w_{j,j-1}$. Let q represent the unit's index in the $(j-1)$ layer and r the unit's index in the jth layer. For notational simplicity, we can ignore the unit's indices in each layer, because the same equation applies (taking careful account of the connection subscripts) to each processing unit. With this convention, we have

$$I_j = \sum_{n_{j-1}=1}^{N_{j-1}} w_{j,j-1} O_{j-1} \tag{D.1b}$$

where n_{j-1} is the index of units in $(j-1)$ layer. Let the total squared error at the output be

$$E_k = \frac{1}{2} \sum_{n_k=1}^{N_k} (r_k - O_k)^2 \tag{D.2}$$

where O_k is the output response and r_k is the desired value (the half factor is introduced to eliminate the constant from the differentiation of D.2. This constant does not affect the final form of the equations). The *Delta rule* or *negative gradient descent rule* is expressed by

$$\Delta w_{k,k-1} = -\alpha \frac{\partial E_k}{\partial w_{k,k-1}} \tag{D.3}$$

where $\alpha > 0$ regulates the magnitude of the correction. It is alternatively called the *learning rate*. The delta rule establishes the condition for weight change given by

$$w_{k,k-1,t} = w_{k,k-1,t-1} - \alpha \frac{\partial E_{k,t}}{\partial w_{k,k-1}} \tag{D.4}$$

where t is the iterate parameter, denoted as a subscript, along with the layer symbols i, j and k. Applying the chain rule for the partial differentiation of (D.3) gives

$$\frac{\partial E_k}{\partial w_{k,k-1}} = \frac{\partial E_k}{\partial I_k} \frac{\partial I_k}{\partial w_{k,k-1}} \tag{C.5}$$

Since

$$\frac{\partial I_k}{\partial w_{k,k-1}} = \frac{\partial \left(\sum_{}^{N_{k-1}} w_{k,k-1}' O_{k-1} \right)}{\partial w_{k,k-1}} = O_{k-1}$$

therefore

$$\Delta w_{k,k-1} = -\alpha \frac{\partial E_k}{\partial w_{k,k-1}} = -\alpha \frac{\partial E_k}{\partial I_k} O_{k-1} = \alpha \delta_k O_{k-1} \tag{D.6}$$

Applying the chain rule to δ_k gives

$$\delta_k = -\frac{\partial E_k}{\partial I_k} = -\frac{\partial E_k}{\partial O_k}\frac{\partial O_k}{\partial I_k} \tag{D.7}$$

From (D.2)

$$\frac{\partial E_k}{\partial O_k} = -(r_k - O_k); \qquad \frac{\partial O_k}{\partial I_k} = \frac{\partial [h_k(I_k)]}{\partial I_k} = h_k'(I_k)$$

where we assume the output O_k is a monotonically increasing function of I_k. So (D.7) becomes

$$\delta_k = (r_k - O_k) h_k'(I_k) \tag{D.8}$$

Substituting (D.8) into (D.6) yields

$$\Delta w_{k,k-1} = \alpha (r_k - O_k) h_k'(I_k) O_{k-1} \tag{D.9}$$

Note that h_k' is the derivative of h_k with respect to I_k, at the output layer k. The weight change in the output layer k is given by (D.9). One might be tempted to apply a similar procedure to obtain the middle layer weight change as

$$\Delta w_{j,j-1} = \alpha (r_j - O_j) h_k'(I_j) O_{j-1}$$

but the problem is we do not know what is the r_j (target values) for the middle layer j. Therefore, to solve the weight change for the middle layer, we need to reformulate $\frac{\partial E_j}{\partial O_j}$ in terms which do not involve $(r_j - O_j)$. Applying (D.7) to the middle layer j we obtain

$$\delta_j = -\frac{\partial E_j}{\partial O_j}\frac{\partial O_j}{\partial I_j} \tag{D.10}$$

$$\frac{\partial O_j}{\partial I_j} = \frac{\partial h_j(I_j)}{\partial I_j} = h_k'(I_j) \tag{D.11}$$

The tricky part of (D.10) is how to represent the first term of the partial derivative. Suppose we use

$$-\frac{\partial E_j}{\partial O_j} = -\sum_{n_{j+1}=1}^{N_{j+1}} \frac{\partial E_j}{\partial I_{j+1}}\frac{\partial I_{j+1}}{\partial O_j} = \sum_{n_{j+1}=1}^{N_{j+1}} \left(-\frac{\partial E_j}{\partial I_{j+1}}\right)\frac{\partial}{\partial O_j}\left(\sum_{n_j=1}^{N_j} w_{j+1,j} O_j\right) \tag{D.12}$$

which involves the differentiation of E_j with respect to the inputs I_{j+1} of the *next* layer, the j+1 layer. Note that the summation symbol appears because of the inputs I_{j+1}. So

$$-\frac{\partial E_j}{\partial O_j} = \sum_{n_{j+1}=1}^{N_{j+1}} \left(-\frac{\partial E_j}{\partial I_{j+1}} \right) w_{j+1,j} = \sum_{n_{j+1}=1}^{N_{j+1}} \delta_{j+1} w_{j+1,j} \tag{D.13}$$

Therefore putting (D.13) and (D.11) into (D.10) yields

$$\delta_j = h_j'(I_j) \sum_{n_{j+1}=1}^{N_{j+1}} \delta_{j+1} w_{j+1,j} \tag{D.14}$$

and the weight change for the middle layer is thus solved as

$$\Delta w_{j,j-1} = \alpha \delta_j O_{j-1} \tag{D.15}$$

where the weight update of any inner layer units can be obtained by propagating the δ_k from the output layer backwards, towards the input layer. This ingenious solution for the learning algorithm, attributable to Rumelhart et al. [1986], is called for this reason, the "backpropagation". It is usual to define the transfer function as

$$h_j(I_j) = \frac{1}{1+\exp[-(I_j + \theta_j)/\theta_0]} \tag{D.16}$$

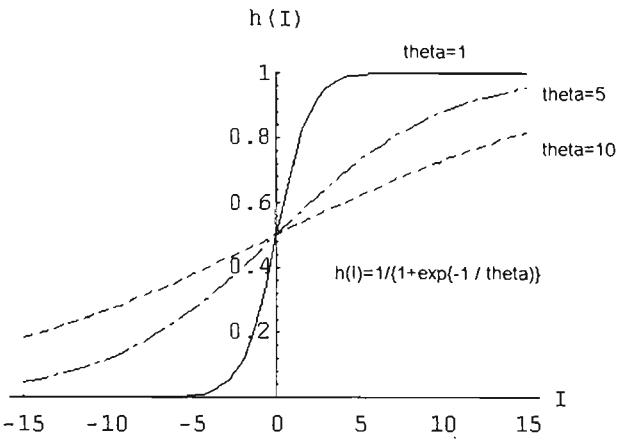


Figure D.2 Graph of the sigmoid function for three values of θ_0. θ_j is assumed zero.

Figure D.2 shows the sigmoid function bounded in the range [0,1] for various inputs I_k. It is seen that θ_0 has a marked effect on the slope of the sigmoid function, and thus the response of the network. θ_0 displaces the function along the I axis of Fig. D.2. Note also that the gradient changes more slowly towards the extrema of the output function, hence a slower convergence by the gradient descent rule. There are ways around this problem, but none is straight forward to use (see Hush and Horn, [1993]). It is easy to show that

$$h_k'(I_j) = O_j(1 - O_j) \tag{D.17}$$

where $O_j = h(I_j)$.

Substituting (D.17) into the preceding results, we may summarise the results of the Delta Learning Law for the backpropagation network, as follows:

(1) Weight change at the output layer is given by

$$\Delta w_{k,k-1} = \alpha \delta_k O_{k-1} \tag{D.18}$$

$$\delta_k = (r_k - O_k) O_k (1 - O_k) \tag{D.19}$$

(2) Weight change at the inner layers is given by

$$\Delta w_{j,j-1} = \alpha \delta_j O_{j-1} \tag{D.20}$$

$$\delta_j = O_j (1 - O_j) \sum_{n_{j+1}=1}^{N_{j+1}} \delta_{j+1} w_{j+1,j} \tag{D.21}$$

or in terms of the output layer

$$\delta_j = O_j (1 - O_j) \sum_{n_k=1}^{N_k} \delta_k w_{k,k-1} \tag{D.22}$$

Appendix E. Derivation of Centre Approximating Equations.

The Centre Approximating (CA) equations facilitate a solution for the Optimum Circle Fit (OCF) equations used in the circle detection algorithm of Chapter 6. This is because OCF does not discriminate cluster points from noise in solving for the circle parameters (centre and radius). The CA equations have superior discrimination of noise from cluster points, but do not predict cluster parameters as accurately as OCF. This is apparent by examining the iterative averaging of cluster points, in the equations (E.8 and E.9) given below.

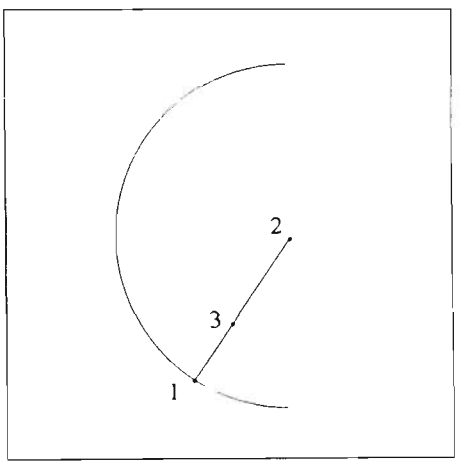


Figure E.1 Circle centre approximation.

Let *point 1* with coordinates $(x_{1,t}, y_{1,t})$ and *point 2* with coordinates $(x_{2,t}, y_{2,t})$ refer to the current cluster point and the current estimated cluster centre, respectively at iteration t. Let *point 3* with coordinates $(x_{2,t-1}, y_{2,t-1})$ represents the previous estimated cluster centre at iteration $t-1$. In estimating the true centre (*point 2*) from the approximate centre (*point 3*) we apply the rule that any line normal to the circle's tangent intersects the circle's centre. Assuming the line containing the points *1*, *2* and *3* is the normal line, the improved centre estimate (*length 12*) is proportional to the previous centre estimate (*length 13*) by a factor

$$h_t = \frac{r_t}{r_{e,t}} \tag{E.1}$$

where r_t = *length 12* and $r_{e,t}$ = *length 13*. The iteration subscript index t indicates that these lengths are successive approximations of the true cluster radius. From Fig. D.1,

$$\Delta E_{x,t} = x_{2,t-1} - x_{1,t} \tag{E.2}$$

$$\Delta E_{y,t} = y_{2,t-1} - y_{1,t} \tag{E.3}$$

$$\Delta X_t = x_{2,t} - x_{1,t} \tag{E.4}$$

$$\Delta Y_t = y_{2,t} - y_{1,t} \tag{E.5}$$

From similar triangles,

$$\Delta X_t = h_t \Delta E_{x,t} \tag{E.6}$$

$$\Delta Y_t = h_t \Delta E_{y,t} \tag{E.7}$$

Expanding the terms of (E.6) and (E.7) with (E.2), (E.3), (E.4) and (E.5) give the desired result

$$x_{2,t} = x_{1,t} + \frac{r_t}{r_{e,t}}(x_{2,t-1} - x_{1,t}) \tag{E.8}$$

$$y_{2,t} = y_{1,t} + \frac{r_t}{r_{e,t}}(y_{2,t-1} - y_{1,t}) \tag{E.9}$$

Initial values of $x_{2,t-1}$ and $y_{2,t-1}$ are assumed zero. Because of the presence of noise, the three points of Fig. E.1 are rarely collinear and unlikely to intersect the circle's centre. To improve the situation, it is necessary to suppress the noise by some sort of averaging process. Provided the cluster points are greater than the noise level, the mean of equations (E.8) or (E.9), calculated from all points in the feature space, will converge towards the centre whereas points remote from the circle effectively cancel out. However, it should be noted that very small values of $r_{e,t}$ can cause overflow error. To avoid this problem, one method is to discard the current computation and use instead, its previous valid result.

Appendix F. Demonstration Program

```
/* Program: VarEta.C
   Author: Paul Im, Dept of Electrical and Electronic Engineering,
           Victoria University of Technology, Australia.
   Date: 26th July, 1996
   Compiler: BorlandC++ 4.5
   Source code: C
   Input data type: ASCII
   Input file: Ruspini.dat (included at the end of program listing)
   Function: Demonstrates the following features described in the
   thesis:
   (1) CPCM framework
   (2) Similarity coefficient of data set s2(j)
   (3) Similarity coefficient of cluster s1
   (4) Variable eta algorithm
   (5) Recluster data from known prototypes
   VarEta is a progressive fuzzy clustering algorithm with a variable
   eta to detect round cluster structures and also reclusters data to
   find more accurate prototypes.
   Display: Cluster points are color coded with '+' symbols. The origin
   of each cluster is represented by a white 'x' in a square. A square
   with a white centre dot represents the centroid tracks. Centroid of
   a cluster is represented by a small white circle. A non-cluster
   point is denoted by a diamond symbol with an inside cross symbol.
*/
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <graphics.h>
#include <time.h>
#include <math.h>
#define bgi_pathname ""
#define pi 3.141592654
#define csize 20  // max number of clusters
#define dsize 75  // max points per cluster
#define nchar 40  // max char length of file name
#define kaput -999 // defines non-cluster point
// Data structure for image points
typedef struct {
   int x,y;
} PINT;
// Data structure for prototype and centroid tracks
typedef struct {
   float x,y;
} PFLT;
// The following are PRIMARY variables
PINT *X; // pointer to image data array
PINT *XC; // duplicate X array for reclustering
PINT *CL[csize]; // pointer to cluster points array
PFLT C[csize]; // prototype array
PFLT v[csize][dsize]; // cluster centroid tracks
unsigned int cs[csize]; // array for number of points in cluster
float cra[csize]; // array for cluster eta circle
unsigned int cf[csize]; // array for cluster iterations
unsigned int n1[csize]; // arrray for cluster tracks count
float vx,vy; // x,y coord prototype variable
float *U; // pointer to point membership
double eta; // eta variable
double etam; // minimum eta variable
double acut; // alphacut variable
double df; // cohesion factor fc
double er; // eta tolerance
double md; // minimum cluster radius
float dcx,dcy; // x,y coord data centroid
double q; // fuzzifier
```

```
unsigned int nc; // number of cluster variable
unsigned int nn; // nearest neighbour variable
unsigned int Na; // count of cluster points
unsigned int mcs; // minimum number of points in a cluster
unsigned int maxx,maxy; // maximum x,y coord dimensions of object
unsigned int iter; // cluster iterations variable
unsigned int tobj,obj; // object total points,current number of points
char dfile[nchar]; // object file name array
float SA[csize]; // cluster similarity coefficient
float SB; // data set similarity coefficient
// The following are SECONDARY variables, in support role
unsigned int newgraph,colr,repeat,cnum;
float cr[csize],cd[csize],ceta[csize];
float B[csize][dsize],A[csize][dsize],S[csize][dsize];
float dx,dy,msv,tpc,sv,svar[csize},CR[csize};
double rmin,nnc,sx,sy,sd,aa;
// Function prototypes
void getinputs(void);
void newcentre(void);
void computeEta(void);
void useNN(void);
void finalcluster(void);
void removepoints(void);
void finalresults(void);
void newcentre(void);
void savedata(void);
void clusterCtr(void);
void newlist(void);
void plus(int,int);
void cross(int,int);
void square(int,int);
void cir(int,int,int);
void diamond(int,int);
void graphIt(void);

main()
{
   FILE *img_file;
   unsigned int rescan,repeat,recluster=1;
   int driver=DETECT,mode;
   int i,j,k;

   initgraph(&driver,&mode,bgi_pathname);
   restorecrtmode();
   printf("VE: Variable eta progressive fuzzy clustering algorithm.\n");
   printf("Data file name > ");
   scanf("%s",dfile);
   if (NULL==(img_file=fopen(dfile,"rt"))) {
      printf("\nError1: Cannot open file %s.\n",dfile);
      exit(1);
   }
   fscanf(img_file,"%d",&obj); // get size of image
   tobj=obj;  // make a duplicate
   if (NULL==(U=(float *)calloc(obj,sizeof(float)))) {
      printf("\nError2: Unable to allocate U memory.\n");
      exit(1);
   }
   if (NULL==(X=(PINT *)calloc(obj,sizeof(PINT)))) {
      printf("\nError3: Unable to allocate X memory.\n");
      exit(1);
   }
   if (NULL==(XC=(PINT *)calloc(obj,sizeof(PINT)))) {
      printf("\nError4: Unable to allocate X memory.\n");
      exit(1);
   }
// Get image data into array X
   for (i=0;i<obj;i++) fscanf(img_file,"%d %d",&X[i].x,&X[i].y);
   fclose(img_file);
// Duplicate image data in XC array
   for (i=0;i<obj;i++) {XC[i].x=X[i].x; XC[i].y=X[i].y;}
// Find max x,y dimensions of image data for display scaling
```

```
   maxx=X[0].x; maxy=X[0].y;
   for (i=1;i<obj;i++){
      maxx=(maxx<X[i].x) ? X[i].x : maxx;
      maxy=(maxy<X[i].y) ? X[i].y : maxy;
   }
// Display unclustered image data
   newgraph=1; graphIt(); newgraph=0;

// Begin CPCM loop
   do { // recluster loop
      getinputs();
      newcentre(); dcx=vx; dcy=vy;
      computeEta();
      nc=0; rescan=1;
      do { // rescan loop
         repeat=1;
// Cluster control loop
      do {  // repeat loop
         useNN();finalcluster();
         if ((Na<mcs) && (obj>=mcs)) {
            removepoints();
            newlist();
            if (obj>0)newcentre();
         }
         else repeat=0;
         if (obj<mcs) {
            repeat=0; rescan=0;
            if (obj>0) {
               for (i=0;i<obj;i++) X[i].x=kaput;
               newlist();
            }
         }
      } while (repeat); // End inner loop
      if (rescan) {
         if (Na>=mcs) {
            savedata();
            newlist();
            if (obj>mcs)newcentre();
            else { // obj<=mcs
               if (obj>0){
                  rescan=0;
                  for (i=0;i<obj;i++) X[i].x=kaput;
                  newlist();
               }
            }
         }
      }
   } while (rescan); // End outer loop
   clusterCtr();
// compute A(i), dist from own centroid
      for (i=0;i<nc;i++)
         for (j=0;j<cs[i];j++) A[i][j]=0;
      for (i=0;i<nc;i++) {
         for (j=0;j<cs[i];j++) {
            dx=CL[i][j].x-C[i].x;
            dy=CL[i][j].y-C[i].y;
            A[i][j]=sqrt(dx*dx+dy*dy);
         }
      }
      msv=0;
      for (i=0;i<nc;i++) CR[i]=0;
      for (i=0;i<nc;i++) {
         aa=0;
         for (j=0;j<cs[i];j++) {
            dx=CL[i][j].x-C[i].x;
            dy=CL[i][j].y-C[i].y;
            aa+=sqrt(dx*dx+dy*dy);
         }
         CR[i]=aa/cs[i];
      }
      for (i=0;i<nc;i++) svar[i]=0;
```

```
     msv=0;
     for (i=0;i<nc;i++) {
        sv=0;
        for (j=0;j<cs[i];j++){
           dx=CL[i][j].x-C[i].x;
           dy=CL[i][j].y-C[i].y;
           sd=sqrt(dx*dx+dy*dy)-CR[i];
           sv+=sd*sd;
           msv+=sd*sd;
        }
        svar[i]=(float)sqrt(sv/cs[i]); // std dev for each cluster
     }
     tpc=0;
     for (i=0;i<nc;i++) tpc+=cs[i]; // total points count
     msv=(float)sqrt(msv/tpc); // std dev for all clusters
// compute B(i), dist from nn of other cluster centroid
     for (i=0;i<nc;i++) {
        for (j=0;j<cs[i];j++) {
           rmin=1E7; cnum=0;
           for (k=0;k<nc;k++) {
              if (k==i)continue; // ensures cluster i != k
              dx=CL[i][j].x-C[k].x; dy=CL[i][j].y-C[k].y;
              nnc=sqrt(dx*dx+dy*dy);
              if (nnc<rmin)  {
                 rmin=nnc;
                 cnum=k;
              }
           }
           dx=CL[i][j].x-C[cnum].x; dy=CL[i][j].y-C[cnum].y;
           B[i][j]=sqrt(dx*dx+dy*dy);
        }
     }
// compute s1(j) and s2 similarity coefficients
     for (i=0;i<nc;i++) {
        for (j=0;j<cs[i];j++) {
           if (B[i][j]>A[i][j]) S[i][j]=1-(A[i][j]/B[i][j]);
           if (B[i][j]<A[i][j]) S[i][j]=(B[i][j]/A[i][j])-1;
           if (B[i][j]==A[i][j]) S[i][j]=0;
        }
     }
     SB=0;
     for (i=0;i<nc;i++) {
        SA[i]=0;
        for (j=0;j<cs[i];j++) {SA[i]+=S[i][j];SB+=S[i][j];}
        SA[i]/=cs[i]; // s1 similarity coefficient for each cluster
     }
     SB/=tpc; // s2 similarity coefficient for data set
  graphIt();
  finalresults();
  for (i=0;i<nc;i++)
     for (j=0;j<n1[i];j++){v[i][j].x=0; v[i][j].y=0;}
  for (i=0;i<nc;i++)n1[i]=0; nc=0;
  for (i=0;i<nc;i++) free(CL[nc]); // free points for reclustering
  printf("Repeat cluster analysis? [0=Exit,1=Repeat] > ");
  scanf("%d",&recluster);
     if (recluster) {
        obj=tobj;
        for (i=0;i<tobj;i++) {
           X[i].x=XC[i].x; X[i].y=XC[i].y;
        }
     }
  } while (recluster); // end of CPCM loop

// free pointers
  free(U);
  free(X); free(XC);
  return 0; // End of program
}

void getinputs(void)
// get user inputs
```

```
{
// printf("Alphacut (normal 0.95) > ");
// scanf("%lf",&acut);
   printf("Cohesion factor (1 to 100) > ");
   scanf("%lf",&df);
   printf("Minimum cluster radius (1 to 100) > ");
   scanf("%lf",&md);
   mcs=5;acut=0.95;er=0.5; // assumed constants
   etam=-md/log(acut); // min eta
}

void newcentre(void)
// find data centroid
{
   int k;
   double sumx,sumy;

   sumx=0; sumy=0;
   for (k=0;k<obj;k++) {
      sumx+=X[k].x;
      sumy+=X[k].y;
   }
   if (obj>0) {
      vx=sumx/obj;  // x-coord data centroid
      vy=sumy/obj;  // y-coord data centroid
   }
   v[nc][n1[nc]].x=vx; v[nc][n1[nc]].y=vy; // save cluster
tracks(centroid)
   n1[nc]++; // counts number of tracks
}

void computeEta(void)
// compute variable eta
// use eta if >= min eta, else use min eta
{
   int k;
   double xc,yc,d;

   d=0;
   for (k=0;k<obj;k++){
      xc=X[k].x-vx;
      yc=X[k].y-vy;
      d+=sqrt(xc*xc+yc*yc);
   }
   if (obj>0){
      eta=df*d/obj;  // eta value
      if (eta<=etam) eta=etam;
   }
}

void useNN(void)
// assign nearest neighbour to prototype
{
   int k;
   double xc,yc,d,dmin;
   nn=0;
   xc=X[0].x-vx; yc=X[0].y-vy;
   dmin=xc*xc+yc*yc;
   for (k=1;k<obj;k++){
      xc=X[k].x-vx; yc=X[k].y-vy;
      d=xc*xc+yc*yc;
      if (d<dmin) {
         dmin=d;
         nn=k; // nearest neighbour point
      }
   }
   vx=X[nn].x; vy=X[nn].y; // assign nearest neighbour to prototype
   v[nc][n1[nc]].x=vx; v[nc][n1[nc]].y=vy; // save cluster
tracks(centroid)
   n1[nc]++; // counts number of tracks
}
```

```
void finalcluster(void)
{
   unsigned int k;
   double pe,err;
   double xc,yc,sux,suy,d;
   pe=0; iter=0;
   do { // do-while loops until eta differential<0.5
// compute memberships
      for (k=0;k<obj;k++) {
         xc=X[k].x-vx;
         yc=X[k].y-vy;
         d=sqrt(xc*xc+yc*yc)/eta; // distance metric
         if (d>25)U[k]=0;
         else U[k]=1/exp(d);  // point membership
      }
      Na=0; sux=0; suy=0;
// find points in cluster
      for (k=0;k<obj;k++) {
         if (U[k]>acut) {
            ++Na; // number of points in cluster
            sux+=X[k].x;
            suy+=X[k].y;
         }
      }
// compute cluster centroids
      if (Na>0) {
         vx=sux/Na;  // x-coord of prototype
         vy=suy/Na;  // y-coord of prototype
         computeEta();
         v[nc][n1[nc]].x=vx; v[nc][n1[nc]].y=vy; // save cluster
tracks(centroid)
         n1[nc]++; // counts tracks
      }

      err=fabs(eta-pe);
      pe=eta;
      iter++;
   } while (err>er); // er=0.5
}

void removepoints(void)
// Remove data points from subsequent clustering
{
   unsigned int k;

   for (k=0;k<obj;k++)
      if (U[k]>acut) X[k].x=kaput;
}

void finalresults(void)
// Show cluster statistics
// CRAD=cluster radius, SDEV=cluster std deviation
// COF=cohesion factor, ACUT=alphacut
// S2=data set sim coeff, S1=cluster sim coeff
{
   unsigned int i,j,k,ncnt;

   for (k=0;k<nc;k++) {
      printf("\nCluster #%d:\n",k+1);
      for (i=0;i<n1[k];i++) {
         printf("%d:%2.1f,%2.1f ",i+1,v[k][i].x,v[k][i].y);
      }
   }
   getch(); printf("\n");
printf("VE:File:%s  Points:%d S2:%6.4f SDEV:%2.2f COF:%1.0f
MCR:%2.1f\n",
           dfile,tobj,SB,msv,df,md);
   printf("Data Centroid:%2.2f,%2.2f ER:%2.1f ACUT:%4.2f MCS:%d\n",
           dcx,dcy,er,acut,mcs);
   for (k=0;k<nc;k++) {
```

```
     printf("  CLUSTER %d [%d points] CTR:%2.2f,%2.2f S1:%6.4f
SDEV:%2.2f\n",
          k+1,cs[k],C[k].x,C[k].y,SA[k],svar[k]);
     printf("CRAD:%2.2f R:%2.2f D:%2.2f ETA:%2.2f ITER:%d\n",
          cra[k],cr[k],cd[k],ceta[k],cf[k]);
     for (i=0;i<cs[k];i++) {
          printf("%d,%d ",CL[k][i].x,CL[k][i].y);
        if ((i+1)%10==0) printf("\n");
     }
     if (cs[k]%10!=0) printf("\n");
     getch();
   }
// Determine noise points
   for (j=0;j<tobj;j++) {X[j].x=XC[j].x; X[j].y=XC[j].y;}
   for (j=0;j<tobj;j++)
      for (k=0;k<nc;k++)
         for (i=0;i<cs[k];i++)
            if (X[j].x>0)
               if (CL[k][i].x==X[j].x&&CL[k][i].y==X[j].y) X[j].x=kaput;
   ncnt=0;
   for (j=0;j<tobj;j++) if (X[j].x>0) ncnt++;
   printf("UNCLUSTERED [%d points]\n",ncnt);
   for (j=0;j<tobj;j++){
      if (X[j].x>0) {
         printf("%d,%d ",X[j].x,X[j].y);
         if ((j+1)%10==0) printf("\n");
      }
   }
   printf("\n");
}

void savedata(void)
// Save cluster statistics
{
   unsigned int k,n;
   double x,y,rsq;
   rsq=0;n=0;
// remove data points from subsequent clustering
   for (k=0;k<obj;k++)
      if (U[k]>acut) {
         x=X[k].x-vx;
         y=X[k].y-vy;
         rsq+=x*x+y*y;
         X[k].x=kaput;n++;
      }

   C[nc].x=vx; // prototype, centroid x coord
   C[nc].y=vy; // prototype, centroid y coord
   cf[nc]=iter; // cluster iterations
   cs[nc]=n;  // number of points in cluster
   v[nc][nl[nc]].x=vx; // x-coord, cluster centroid tracks
   v[nc][nl[nc]].y=vy; // y-coord, cluster centroid tracks
   nl[nc]++;
   cra[nc]=-eta*log(acut); // cluster eta circle
   ceta[nc]=eta;
   cr[nc]=sqrt(rsq/(n-1)); // cluster rms radii
   cd[nc]=100*(n-1)/(pi*rsq/(n-1));  // cluster density
   ++nc;
}

void clusterCtr(void)
// Re-cluster from known prototypes
// to obtain more accurate clusters
{
   unsigned int i,k,np;
   double xc,yc,r,sx,sy;

   for (i=0;i<nc;i++){
      if (NULL==(CL[i]=(PINT *)calloc(100,sizeof(PINT)))) {
         printf("\nError5: Unable to allocate CL memory.\n");
         exit(1);
```

```
      }
   }
   for (i=0;i<tobj;i++) {X[i].x=XC[i].x; X[i].y=XC[i].y;}
   for (i=0;i<nc;i++) {
      np=0; sx=sy=0;
      for (k=0;k<tobj;k++){
         if (X[k].x>0) {
            xc=X[k].x-C[i].x; yc=X[k].y-C[i].y;
            r=sqrt(xc*xc+yc*yc);
            if (r<=cra[i]) {
// save improved cluster statistics
               CL[i][np].x=X[k].x;  // x-coord cluster points
               CL[i][np].y=X[k].y;  // y-coord cluster points
               sx+=X[k].x; sy+=X[k].y;
               np++; X[k].x=kaput;  // remove cluster points
            }
         }

      }
      if (np>0){vx=sx/np; vy=sy/np;} // improved prototype positions
      C[i].x=vx; C[i].y=vy;  // save improved prototype positions
      cs[i]=np;  // save number of cluster points
   }
}

void newlist(void)
// Update data array list and fills up
// vacant array cells with image point data
{
   unsigned int k,ind=0;

   for (k=0;k<obj;k++) {
      if (X[k].x==kaput) continue;
      else {
         X[ind].x=X[k].x;
         X[ind].y=X[k].y;
         ind++;
      }
   }
   obj=ind;
}

void plus(int x,int y)
// Constructs a plus symbol
{
   setcolor(colr);
   line(x-3,y,x+3,y); line(x,y-3,x,y+3);
}

void cross(int x,int y)
// Constructs a cross symbol
{
   setcolor(colr);
   line(x-2,y-2,x+2,y+2); line(x+2,y-2,x-2,y+2);
}

void cir(int x,int y,int r)
// Constructs a circle symbol
{
   setcolor(colr);
   circle(x,y,r);
}

void diamond(int x,int y)
// Constructs a diamond symbol
{
   int n=5;
   setcolor(colr);
   line(x,y-n,x-n,y);
   line(x-n,y,x,y+n);
   line(x,y+n,x+n,y);
```

```
   line(x+n,y,x,y-n);
}

void square(int x,int y)
// Constructs a square symbol
{
   int n=3;
   setcolor(colr);
   line(x-n,y-n,x+n,y-n);
   line(x+n,y-n,x+n,y+n);
   line(x+n,y+n,x-n,y+n);
   line(x-n,y+n,x-n,y-n);
}

void graphIt(void)
// Display color coded cluster points, origins,
// centroid tracks and unclustered points (if any)
{
   int i,j,k;
   unsigned int xp,yp,icnt,xo,yo,xw,yw;
   unsigned char ints[5],ni,nih;
   double xs,ys;

   setgraphmode(getgraphmode()); // restore graphics mode
   setcolor(15);
   xo=100;yo=10;xw=yw=440; // (xo,yo)=origin;xw,yw are x,y coords widths
// Get x,y coordinates scale factor
   if (maxx<50||maxy<50){xs=(double)xw/maxx-3;ys=xs;}
   else {xs=(double)xw/maxx-0.1; ys=(double)yw/maxy-0.1;}
   if(maxx>maxy)ys=xs;
   else xs=ys;
   if (maxx>200||maxy>200){ni=30;nih=15;}
   else if (maxx>100||maxy>100){ni=20;nih=10;}
   else if (maxx>50||maxy>50){ni=10;nih=5;}
   else {ni=1;nih=1;}
   rectangle(xo,yo,xo+xw,yo+yw);
   line(xo,yo+yw+2,xo,yo+yw+7); // zero x tick
   line(xo-2,yo+yw,xo-7,yo+yw); // zero y tick
   itoa(0,ints,10); moveto(xo-4,yo+yw+12); outtext(ints); // x=0
   itoa(0,ints,10); moveto(xo-32,yo+yw-3); outtext(ints); // y=0
   for (icnt=1,k=1;k<(xw+1);k++) { // x coord ticks and number labels
      if (k%((int)(nih*xs+0.5))==0)
        {xp=xo+k;yp=yo+yw+2;line(xp,yp,xp,yp+5);}
      if (k%((int)(ni*xs+0.5))==0) {
        {itoa(ni*icnt++,ints,10); moveto(xp-4,yp+10); outtext(ints);}
   }
   for (icnt=1,k=1;k<(yw+1);k++) {  // y coord ticks and number labels
      if (k%((int)(nih*ys+0.5))==0)
        {xp=xo-2;yp=yo+yw-k;line(xp,yp,xp-5,yp);}
      if (k%((int)(ni*ys+0.5))==0)
         {itoa(ni*icnt++,ints,10); moveto(xp-32,yp-3); outtext(ints);}
   }
   if (newgraph){
      colr=11;
      for (k=0;k<obj;k++) {
         xp=xo+xs*X[k].x; yp=yo+yw-ys*X[k].y;
         cross(xp,yp);
      }
   }
   else {
      colr=15;
// A square with inside white cross represents cluster origin
// Color of square corresponds to color of cluster
// Cross (white) symbol denotes cluster origin
      for (k=0;k<nc;k++) {
         xp=xo+xs*v[k][0].x+0.5; yp=yo+yw-ys*v[k][0].y+0.5;
         cross(xp,yp);
      }
      colr=2;
// Square (colored) symbol denotes cluster origin
      for (k=0;k<nc;k++) {
```

```
        xp=xo+xs*v[k][0].x+0.5; yp=yo+yw-ys*v[k][0].y+0.5;
        square(xp,yp);
        colr++;
      }
      colr=2;
// Plus symbol denotes cluster point
      for (k=0;k<nc;k++) {
        for (i=0;i<cs[k];i++) {
          xp=xo+xs*CL[k][i].x+0.5; yp=yo+yw-ys*CL[k][i].y+0.5;
          plus(xp,yp);
        }
        xp=xo+xs*C[k].x+0.5; yp=yo+yw-ys*C[k].y+0.5;
// Show variable eta circle of cluster
// Color of circle corresponds to color of cluster
        cir(xp,yp,(xs*cra[k]+0.5));
        colr++;
      }
// Remove cluster points for later noise points identification
      for (j=0;j<tobj;j++) {X[j].x=XC[j].x; X[j].y=XC[j].y;}
      for (j=0;j<tobj;j++)
        for (k=0;k<nc;k++)
          for (i=0;i<cs[k];i++) {
            if (X[j].x>0) {
              if (CL[k][i].x==X[j].x&&CL[k][i].y==X[j].y)
                X[j].x=kaput;
            }
          }
// Diamond and cross symbols denote unclustered point
      colr=7;
      for (j=0;j<tobj;j++)
        if (X[j].x>0) {
          diamond(xo+xs*X[j].x+0.5,yo+yw-ys*X[j].y+0.5);
          cross(xo+xs*X[j].x+0.5,yo+yw-ys*X[j].y+0.5);
        }
// Square and a centre white dot symbols denote cluster tracks
      colr=2;
      for (k=0;k<nc;k++) {
        for (i=1;i<n1[k];i++) {
          xp=xo+xs*v[k][i].x+0.5; yp=yo+yw-ys*v[k][i].y+0.5;
          square(xp,yp);
          putpixel(xp,yp,15);
        }
        colr++;
      }
      colr=15;
// Small circle symbol denotes cluster prototype
      for (k=0;k<nc;k++) {
        xp=xo+xs*C[k].x+0.5; yp=yo+yw-ys*C[k].y+0.5;
        cir(xp,yp,2); // cluster prototype
      }
    }
    getch();
    restorecrtmode();
}
```

Data File: Ruspini.dat

```
75
4 53 5 63 10 59 9 77 13 49 13 69 12 88 15 75 18 61 19 65
22 74 27 72 28 76 24 58 27 55 28 60 30 52 31 60 32 61 36 72
28 147 32 149 35 153 33 154 38 151 41 150 38 145 38 143 32 143 34 141
44 156 44 149 44 143 46 142 47 149 49 152 50 142 53 144 52 152 55 155
54 124 60 136 63 139 86 132 85 115 85 96 78 94 74 96 97 122 98 116
98 124 99 119 99 128 101 115 108 111 110 111 108 116 111 126 115 117 117 115
70 4 77 12 83 21 61 15 69 15 78 16 66 18 58 13 64 20 69 21
66 23 61 25 76 27 72 31 64 30
```

Glossary

This glossary is intended to present a succinct explanation or clarification of some unfamiliar pattern recognition terms or concepts used in the thesis. The items in the glossary represent the more significant terms alluded to directly or indirectly in the thesis, without attempting to be comprehensive. Works from the following authors, where cited in the glossary, are acknowledged with capitalised characters enclosed in square brackets eg. [HS]. Unacknowledged items are from the author.

BF = [Burden and Faires, 1989]
FS = [Freeman and Skapura, 1992]
GW = [Gonzalez and Woods, 1992]
HS = [Haralick and Shapiro, 1993]
JD = [Jain and Dubes, 1988]
JKS = [Jain, Kasturi and Schuck, 1995]
SHB = [Sonka, Hlavac and Boyle, 1993]

Accuracy
Accuracy refers to the degree of closeness an estimate has to the true value of what it is estimating. [HS]

Area analysis, region analysis
In area analysis, the area of the image containing the objects or entities to be processed is located by some simple algorithm. A more complex processing algorithm is applied only in the located area. This strategy of processing can often increase execution speed. The algorithm locating the area to be processed is called the *focus-of-attention mechanism*. [HS]

Bayes decision rule
A Bayes decision rule is one that treats the units independently and assigns a unit u having pattern measurements or features d to the category c whose conditional probability, given d, is highest. [HS] See decision rule.

Binary image
A binary image is an image in which each pixel takes either the value zero or non-zero. Usually the non-zero value is assumed to be 1. [HS] See pixel.

Blob or connected component
A blob is a maximal-sized connected region. [HS] See blob analysis.

Blob analysis, connectivity analysis, connected component analysis
In blob analysis, the position and shape properties of each connected component are measured. Typical shape properties include area, perimeter, number of holes, bounding rectan-

gle, extremal points, centroid, second moments, and orientation derived from second moments or extremal points. The connected components are then identified or classified by a decision rule on the basis of their measured properties. [HS] See blob, area analysis.

Classifier
A classifier is a device or process that sorts patterns into categories or classes. [HS]

Categorised
See classified

Classified, identified, recognised, categorised
A unit is said to be classified if the decision rule is able to assign it to some category from the set of given categories. In some applications there may be a definite distinction between recognise and identify. In these applications, for a unit to be recognised, the decision rule must be able to assign it to a type of category that includes many sub-categories. For a unit to be identified, the decision rule must be able to assign it not only to a type of category but also to a sub-category of the category type. For example, a small area ground patch that may be recognised as containing trees may be specifically identified as containing apple trees. [HS]

Classification, identification
Refers to the class or category assignment of data. A classified object is said to be identified. See classified.

Cluster
A cluster is a homogenous group of units that are very "like" one another. "Likeness" between units is usually determined by the association, similarity or distance between the measurement patterns associated with the units. [HS] See cluster assignment function.

Cluster prototype
A cluster prototype represents or characterises a cluster in some way eg. the centre or centroid of the cluster. In fuzzy clustering, it is obtained by the iterative minimization of an objective function J involving the fuzzy memberships.

Cluster assignment function
A cluster assignment function is a function that assigns each observed unit to a cluster on the basis of their corresponding features. Sometimes the units are treated independently. In this case the cluster assignment function can be considered as a transformation from measurement space to a set of clusters. [HS]

Clustering, cluster analysis, pattern classification
Cluster analysis is defined as the formal study of algorithms and methods for grouping or classifying objects. [JD].

Clustering is concerned with constructing the cluster assignment function that groups similar units. Clustering is synonymous with *pattern classification* or *numerical taxonomy*. [HS]

Computer vision
Computer vision is the combination of image processing, pattern recognition, and artificial intelligence technologies that focuses on the computer analysis of one or more images, taken with a single/multiband sensor in time sequence. The analysis recognises, locates the

position and orientation of, and provides a sufficiently detailed symbolic description or recognition of those imaged objects deemed to be of interest in the three dimensional environment. The computer vision process often uses geometric modeling and complex-knowledge representations in an expectation- or model-based matching and searching methodology. The searching can include bottom-up, top-down, blackboard, hierarchical, and heterarchical control strategies. [HS] See machine vision systems.

Connectivity analysis, connected component analysis
See blob analysis.

Data
Data is any qualitative or quantitative information containing useful or meaningful patterns or structure. In machine vision, meaningful data is typically embedded in noise or a complex surrounding. Image processing or clustering algorithms are processes that can extract meaningful patterns from noise or the complex surrounding and give an interpretation of the extracted data. This data is called an object or objects of interest.

Decision rule or simple decision rule
A decision rule usually assigns one and only one category to each observed unit on the basis of the sequence of measurement patterns in the data sequence S_d or on the basis of the corresponding sequence of feature patterns. A simple decision rule f is a decision rule that assigns a category to a unit solely on the basis of the measurements or features associated with the unit. Hence the units are treated independently, and the decision rule f may be thought of as a function that assigns one and only one category to each pattern in measurement space or to each feature in feature space. [HS]

Detect, detection
A unit is said to be detected if the decision rule is able to assign it as belonging only to some given subset A of categories from the set C of categories. To detect a unit does not imply that the decision rule is able to identify the unit as specifically belonging to one particular category. [HS]

Digital image
A digital image or digitised image is an image in digital format obtained by partitioning the area of an image into a finite two dimensional array of small, uniformly shaped, mutually exclusive regions called resolution cells and assigning a representative image value to each such spatial region. A digital image may be abstractly thought of as a function whose domain is a finite two dimensional set of resolution cells and whose range is the set of possible image intensities. [HS]

Discriminant function, linear discriminant function
A discriminant function $f_i(d)$ is a scalar function whose domain s usually measurement space and whose range is usually the real numbers. When $f_i(d) \geq f_k(d)$, for $k = 1,2,...,K$, then the decision rule assigns the ith category to the unit giving rise to pattern d. A linear discriminant function f is a discriminant function of the form

$f(d) = \sum_{j=1}^{n} a_j \delta_j + a_0$, where $d = (\delta_1, \delta_2, ..., \delta_n)$ represents the measurement pattern.

[HS]

Feature, feature pattern, feature vector or pattern feature
A feature is an N-tuples or vector whose components are functions of the initial measurement pattern variables or some subset of them. Feature N-tuples or vectors are designed to

contain a high amount of information relative to the discrimination between units of the types of categories in the given category set. Sometimes the features are predetermined; at other times they are determined when the pattern discrimination problem is being solved. In image pattern recognition, features often contain information relative to gray tone intensity, texture, or region shape. [HS] See feature space.

Feature space
A feature space is the set of all possible feature N-tuples. [HS]

Generalisation
In the context of neural networks, it refers to the capability of the network to associate key features of new input vectors to the features of the training class. Generalisation capabilities are related to neural network topology, the representativeness of the training samples and the number of training patterns. [FS]

Gray scale image or gray level image
A gray scale image is an image in which each pixel has a value in a range larger than just 0 or 1. Gray scale images typically have values in the range 0 to 63, 0 to 255, or 0 to 1,023 corresponding to 6-bit, 8-bit or 10-bit digitisations. [HS]

Histogram or histogram image
A histogram is a function defined on the set of image intensity values of non-negative integers. The value $h(k)$ is given by the number of pixels in the image having image intensity k. For images having a large gray tone range, the image will often be quantised before being histogrammed or will be quantised on the fly during the histogramming process. [HS]

Hyperplane
A hyperplane is a decision boundary which arises from the use of affine discriminant functions.

Identified
See classified.

Identification
See classification.

Illumination
The illumination at a point on a surface is the luminous flux incident on an infinitesimal element of the surface centred at the given point divided by area of the surface element. The unit of illumination is the *lux* or meter candle, being equivalent to one lumen per square meter. The illumination at a point on a surface due to a point source of light is proportional to the luminous intensity of the source in the direction of the surface point and to the cosine of the angle between this direction and surface normal direction. It is inversely proportional to the square of the distance between the surface point and the source. [HS]

Image
An image is a spatial representation of an object contained in a two or three dimensional scene. In computer or machine vision, "image" usually means recorded image such as a video image, digital image, or a picture. In a two dimensional case, it may be thought of as a continuous function I of two variables in a rectangular plane or region with values at

spatial or orthogonal coordinates (r, c) denoted by $I(r, c)$. [HS] See video image, digital image, image intensity, gray scale image, image processing and histogram.

Image processing
Image processing encompasses all the various operations that can be applied to image data. These include, but are not limited to, contrast stretching, edge enhancement, image enhancement, preprocessing, quantisation, spatial filtering, matching and recognition techniques. [GW] See image.

Interpretation
Interpretation involves assigning meaning to an ensemble of recognised objects. [GW]

Labelled
An object is said to be labelled if its identity or class membership is known. For example, an approximately circular silhouette can represent any number of objects such as a ball, egg, orange or balloon. However, given additional information that the object has the smell, texture, color and weight of an orange, the object is said to be identified as an orange.

Linearly separable
Two classes are said to be separable if their class regions do not overlap. If for every class region there exists a hyperplane that separates it from all other class regions, the classes are said to be linearly separable. [HS] See hyperplane.

Machine vision system
A machine vision system is a system capable of acquiring one or more images of an object; of processing, analysing and measuring various characteristics of the acquired images; and of interpreting the results of the measurements in such a way that some useful decision can be made about the object. Functions of machine vision systems include locating, inspecting, gauging, identifying, recognising, counting, and motion estimating. Also see computing vision. [HS]

Measurement space
A measurement space is a set large enough to include the set of all possible measurement patterns that could be obtained by observing some set of units. [HS]

Measurement vector
A measurement vector is the ordered N-tuples of measurements obtained from a unit under observation. Each component of the N-tuples is a measurement of a particular quality, feature, or characteristic of the unit. In image pattern recognition, the units are usually picture elements or simple formations of picture elements, and the measurement N-tuples are the corresponding gray tone intensities, gray tone intensity N-tuples, or properties of formations of gray tone intensities. [HS]

Neural network, perceptron
A neural network is an interconnected network of non-linear units or processing elements capable of learning and self-organising. The response of a unit or a processing element is a non-linear monotonic function of a weighted sum of the inputs to the processing elements. The weights, called *synaptic weights*, are modified by learning or reinforcement algorithm. Typical non-linear processing functions are $\mathrm{sgn}(x)$, $1/(1+e^{-x})$, and $\tanh(x)$. When each processing element contributes one component to the output response vector, the percep-

tron is called a simple perceptron. Processing units whose outputs only indirectly influences the components of the output response vector are called *hidden units*. [HS]

The attraction of neural networks for pattern recognition lies in their ability to partition the feature space using non-linear boundaries for classes. Neural networks are limited in their ability to introduce known facts about the application domain and the difficulty in debugging performance. [JKS]

Noise

Two types of noise may be identified.

1. The first kind is random variations in the intensity values. Some of the common types of noise belonging to this category are: (i) salt and pepper, (ii) impulse and (iii) Gaussian. Salt and pepper noise contain random occurrences of both black and white intensity values. Impulse noise contains only white intensity values. Gaussian noise contains variations in intensity that are drawn from a Gaussian or normal distribution. [JKS]
2. The other kind of noise is associated with the by products of image processing or from a clustering algorithm. For example blurred edges from an edge operator and low membership values from a fuzzy clustering algorithm.

Norm, vector norm

Let $\Re^n$ denotes the set of all n-dimensional column vectors with real number coefficients. To define a distance in $\Re^n$, we use the notion of a norm. A vector norm on $\Re^n$ is a function, $\|\cdot\|$, from $\Re^n$ into $\Re$ with the following properties:

(i) $\|\mathbf{x}\| \geq 0$ for all $\mathrm{x} \in \Re^n$,

(ii) $\|\mathbf{x}\| = 0$ if and only if $\mathbf{x} = [0,0,\ldots,0]^T \equiv \mathbf{0}$,

(iii) $\|\alpha\mathbf{x}\| = |\alpha| \, \|\mathbf{x}\|$ for all $\alpha \in \Re$, and $\mathbf{x} \in \Re^n$,

(iv) $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$ for all $\mathbf{x}, \mathbf{y} \in \Re^n$.

where $\mathbf{x} = [x_1,x_2,\ldots,x_n]^T$ is the vector in $\Re^n$. [BF] See Appendix A.

Optimal, optimality

Optimality in the context of clustering is understood in two senses. As subjective optimality, it refers to an ideal or natural perception of grouping in data. Objective optimality is restricted to the notion of objective performance measures such as the least squares error, cluster volume or cluster shape, or other cluster parameters. See optimisation techniques.

Optimisation techniques

Optimisation implies the notion of finding the best fit of an objective function to data. A function optimisation may be defined as follows: given some finite domain D and a function f: $D \rightarrow R$, R being the set of real numbers, find the best value in D under f. Finding the best value in D is understood as finding a value $x \in D$ yielding either the minimum (function minimisation) or the maximum (function maximisation) of the function f:

$$f_{\min}(x) = \min_{x \in D} f(x), \qquad f_{\max}(x) = \max_{x \in D} f(x)$$

The function f is called the objective function. The design of the objective function is a key factor in the performance of any optimisation algorithm. Conventional approaches to optimisation use calculus based methods which can be compared to hill-climbing (in maximisation problem) or gradient descent (in minimisation problem) – the gradient of the objective function gives the steepest direction of climb or descent. The main limitation of the calculus based methods is their local behaviour; for example, the search can easily end in a local minimum and the global minimum can be missed. There are several methods to improve the search for a global minimum, like dynamic programming, random searches and genetic algorithms. [SHB]

Partition
A partition is a cluster or natural grouping of data resulting from a clustering process.

Pattern, measurement pattern, pattern vectors
A pattern is the data structure of the measurements resulting from observing a unit. The word pattern is used in three distinct senses: as measurement pattern; as feature pattern; and as the dependency pattern or patterns of relationships among the components of any measurement or feature N-tuples derived from units of a particular category and that are unique to those N-tuples, that is, they are dependencies that do not occur in any other category. [HS]

A pattern vector is represented by bold face letter such as $\mathbf{x} = (x_1, x_2 \ldots, x_n)^T$, where each component x_i represents the ith descriptor and n is the number of features or dimensions of such descriptors. The nature of the components of a pattern vector $\mathbf{x}$ depends on the measurement technique used to describe the physical pattern. [GW]

Pattern recognition
Pattern recognition techniques can be used to construct decision rules that enable one to identify units on the basis of their measurement patterns. Pattern recognition techniques can also be employed to cluster units having similar enough measurement patterns. In statistical pattern recognition, the measurement patterns have the form of N-tuples or vectors. In *syntactic pattern recognition*, the measurement patterns have the form of sentences from the language of a phrase structure grammar. In *structural pattern recognition*, the measurements do not have the form of an N-tuples or vector. Rather, the unit being measured is encoded in terms of its parts and their relationships and properties. Also see units. [HS]

Pattern recognition is a high level image processing activity involving the nature of data interpretation, and the application of constraints and idealisations to assist this task, such as by the reduction of task complexity or by optimisation techniques. Methods of pattern recognition may be classified into: (i) decision-theoretic (such as Bayes classifier, neural networks and fuzzy clustering), (ii) structural (such strings and trees) and (ii) image interpretation (such as predicate logic, semantic networks and production systems). [GW] See recognition.

Pixel, picture element or pel
A pixel is a pair whose first member is a resolution cell or (row,column) spatial position and whose second member is the image intensity value or vector of image values associated with the spatial position. Also see resolution cell. [HS]

Point operator
A point operator is an image operator in which the output image value at each pixel position depends only on the input image value at the corresponding pixel position. [HS]

Precision
Precision refers to the degree of closeness an estimate has to its expected value. [HS]

Preprocessing
Preprocessing is an operation applied before pattern identification is performed. Preprocessing produces, for the categories of interest, pattern features that tend to be invariant under changes such as translation, rotation, scale, illumination level, and noise. In essence, preprocessing converts the measurement patterns to a form that allows a simplification in

the decision rule. Preprocessing can enhance images, segment target patterns and detect, normalise or centre objects of interest. [HS]

Prototype or prototype pattern
A prototype is the observable or characteristic measurement or feature pattern derived from units of a particular category. A category is said to have a prototype pattern only if the characteristic pattern is highly representative of the N-tuples obtained from units of that category. [HS] See cluster prototype.

Recognised
See classified.

Recognition
Recognition is the process that assigns a label to an object based on the information derived from quantitative and qualitative features of the object. [GW] See pattern recognition.

Region
Refers to an area description of connected pixels in an image.

Resolution
Resolution is a generic term that describes how well a system process, component material, or image can reproduce an isolated object consisting of separate closely spaced objects or lines. [HS]

Resolution cell
A resolution cell is the smallest, most elementary constituent by area, having an associated image intensity in a digital image. A resolution cell is referenced by its spatial coordinates which are the centre coordinates of its area. The resolution cell or spatial formations of resolution cells constitute the basic unit for low level processing of digital image data. Resolution cells usually have areas that are square, rectangular or hexagonal. [HS]

Robust, robustness
A vision procedure is said to be robust or possesses robustness if small changes in the assumed model on which the procedure or technique was developed produce only small changes in the result. Small fractions of the data that do not fit the assumed model, and in fact are very far from fitting it, constitute a small change in the assumed model. Data not fitting an assumed model may be due to rounding or quantising errors, gross errors, or the fact that the model itself is an idealised approximation of reality. [HS]

In the context of clustering algorithms, robustness is a property that is associated with a resistance (to varying degrees) to unwanted effects due to such factors as the presence of or interference from noise or other data, nature of data processing and initial conditions of the cluster parameters. In neural networks, robustness means the essential preservation of generalisation attributes despite the inexactness of the input data to training data.

Segmentation
Segmentation is a process that typically partitions the spatial domain of an image into mutually exclusive subsets called regions. Each region is uniform and homogeneous with respect to some property, such a tone or texture, and its property value differs in some significant way from that of each neighbouring region. An image segmentation process that

uses image intensity as a property value produces regions that are called discrete tonal features. [HS]

Structure
Refers to meaningful patterns in data in contrast to random patterns or noise.

Template matching
Template matching is an operation that can be used to find out how well a template subimage matches a window of a given image. The degree of matching is often determined by translating the template subimage all over the given image and for each position, evaluating the cross-correlation or the sum of the squared or absolute image intensity differences of corresponding pixels. Template matching can also be used to best match an observed measurement pattern with a prototype pattern. [HS]

Thresholding, multilevel thresholding
Thresholding is an image point operation that produces a binary image from a gray scale image. A binary-1 is produced on the output image whenever a pixel value on the output image is above a specified minimum threshold level. A binary-0 is produced otherwise. Alternately, thresholding can produce a binary-1 on the output image whenever a pixel value on the input image is below a specified maximum level. A binary-0 is produced otherwise. Multilevel thresholding is a point operator employing two or more thresholds. Pixel values that are in the interval between two successive threshold values are assigned an index associated with the interval. Also see point operator. [HS]

Units
The unit is the entity that is observed and whose measured properties constitute the measurement pattern. The simplest and most practical unit to observe and measure in the pattern recognition of image data is often the pixel (the gray tone intensity or the gray tone intensity N-tuples in a particular resolution cell). This is what makes pictorial pattern recognition so difficult, because the objects requiring analysis or identification are not single pixels but are often complex spatial formations of pixels. [HS]

Vector norm
See norm and Appendix A.

Video image
A video image is an image in electronic signal format capable of being displayed on a cathode ray tube, screen or monitor. The video signal can be generated from devices like a CCD camera, a vidicon, a flying spot scanner, a tactile sensor, a range sensor or a frame buffer driving a digital-to-analog convertor. [HS]