FUZZY MODELS AND ALGORITHMS FOR PATTERN RECOGNITION AND IMAGE PROCESSING

THE HANDBOOKS OF FUZZY SETS SERIES

Series Editors
Didier Dubois and Henri Prade
IRIT, Université Paul Sabatier, Toulouse, France

FUNDAMENTALS OF FUZZY SETS, edited by Didier Dubois and Henri Prade
MATHEMATICS OF FUZZY SETS: ***Logic, Topology, and Measure Theory***, edited by Ulrich Höhle and Stephen Ernest Rodabaugh
FUZZY SETS IN APPROXIMATE REASONING AND INFORMATION SYSTEMS, edited by James C. Bezdek, Didier Dubois and Henri Prade
FUZZY MODELS AND ALGORITHMS FOR PATTERN RECOGNITION AND IMAGE PROCESSING, by James C. Bezdek, James Keller, Raghu Krisnapuram and Nikhil R. Pal
FUZZY SETS IN DECISION ANALYSIS, OPERATIONS RESEARCH AND STATISTICS, edited by Roman Slowinski
FUZZY SYSTEMS: ***Modeling and Control***, edited by Hung T. Nguyen and Michio Sugeno
PRACTICAL APPLICATIONS OF FUZZY TECHNOLOGIES, edited by Hans-Jürgen Zimmermann

FUZZY MODELS AND ALGORITHMS FOR PATTERN RECOGNITION AND IMAGE PROCESSING

James C. Bezdek
University of West Florida

James Keller
University of Missouri

Raghu Krisnapuram
Colorado School of Mines

Nikhil R. Pal
Indian Statistical Institute

Library of Congress Cataloging-in-Publication Data

Fuzzy models and algorithms for pattern recognition and image processing / James C. Bezdek ... [et al.].
p. cm. – (The handbooks of fuzzy sets series)
Includes bibliographical references and index.
ISBN 0-387-24515-4 (softcover : alk. paper)
ISBN 0-7923-8521-7 (hardcover) © 1999 Kluwer Academic Publishers
1. Optical pattern recognition. 2. Fuzzy algorithms. 3. Cluster analysis. 4. Image processing. 5. Computer vision. I. Bezdek, James C., 1939- II. Series.

TA1650.F89 2005
006.4'2--dc22

2005042541

Printed in the United States of America.

9 8 7 6 5 4 3 2 1 SPIN 11384601

springeronline.com

Contents

Series Foreword

Fuzzy sets were introduced in 1965 by Lotfi Zadeh with a view to reconcile mathematical modeling and human knowledge in the engineering sciences. Since then, a considerable body of literature has blossomed around the concept of fuzzy sets in an incredibly wide range of areas, from mathematics and logic to traditional and advanced engineering methodologies (from civil engineering to computational intelligence). Applications are found in many contexts, from medicine to finance, from human factors to consumer products, from vehicle control to computational linguistics, and so on.... Fuzzy logic is now used in the industrial practice of advanced information technology.

As a consequence of this trend, the number of conferences and publications on fuzzy logic has grown exponentially, and it becomes very difficult for students, newcomers, and even scientists already familiar with some aspects of fuzzy sets, to find their way in the maze of fuzzy papers. Notwithstanding circumstantial edited volumes, numerous fuzzy books have appeared, but, if we except very few comprehensive balanced textbooks, they are either very specialized monographs, or remain at a rather superficial level. Some are even misleading, conveying more ideology and unsustained claims than actual scientific contents.

What is missing is an organized set of detailed guidebooks to the relevant literature, that help the students and the newcoming scientist, having some preliminary knowledge of fuzzy sets, get deeper in the field without wasting time, by being guided right away in the heart of the literature relevant for her or his purpose. The ambition of the HANDBOOKS OF FUZZY SETS is to address this need. It will offer, in the compass of several volumes, a full picture of the current state of the art, in terms of the basic concepts, the mathematical developments, and the engineering methodologies that exploit the concept of fuzzy sets.

This collection will propose a series of volumes that aim at becoming a useful source of reference for all those, from graduate students to senior researchers, from pure mathematicians to industrial information engineers as well as life, human and social sciences scholars, interested in or working with fuzzy sets. The original feature of these volumes is that each chapter - except in the case of this volume, which was written entirely by the four authors - is written by one or several experts in the topic concerned. It provides an introduction to the topic, outlines its development, presents the major results, and supplies an extensive bibliography for further reading.

The core set of volumes are respectively devoted to fundamentals of fuzzy sets, mathematics of fuzzy sets, approximate reasoning and information systems, fuzzy models for pattern recognition and image processing, fuzzy sets in decision research and statistics, fuzzy systems in modeling and control, and a guide to practical applications of fuzzy technologies.

D. Dubois H. Prade
Toulouse

The authors Rather than compile many chapters written by various authors who use different notations and semantic descriptions for the same models, we decided to have a small team of four persons write the entire volume. Each of us assumed the role of lead author for one or more of the chapters, and the other authors acted like consultants to the lead author. Each of us helped the lead author by contributing examples, references, diagrams or text here and there; and we all reviewed the entire volume three times. Whether this approach was successful remains to be seen.

The plan What we tried to do is this: identify the important work that has been done in fuzzy pattern recognition, describe it, analyze it, and illustrate it with examples that an interested reader can follow. As with all projects of this kind, the material inevitably reflects some bias on the part of its authors (after all, the easiest examples to give already live in our own computers). Moreover, this has become an enormous field, and the truth is that it is now far too large for us to even *know about* many important and useful papers that go unrecognized here. We apologize for our bias and our ignorance, and accept any and all blame for errors of fact and/or omission. How current is the material in the book? Knuth (1968) stated that "It is generally very difficult to keep up with a field that is economically profitable, and so it is only natural to expect that many of the techniques described here eventually be superseded by better ones". We cannot say it better.

The numbering system The atomic unit for the numbering system is the chapter. Figures, tables, examples and equations are all numbered consecutively within each chapter. For example, Figure 3.5 is Figure 5 of Chapter 3. The beginning and end of examples are enclosed by goofy looking brackets, like this:

Example 5.4 Did you ever have to finally decide? To pick up on one and let the other one ride, so many changes,.....

The algorithms: art, science and voodoo There are a lot of algorithms in the book. We ran many, but not certainly not all, of the experiments ourselves. We have given pseudo code for quite a few algorithms, and it is really pseudo in the sense that it is a mixture of three or four programming languages and writing styles. Our intent is to maximize clarity and minimize dependence on a particular language, operating system, compiler, host platform, and so on. We hope you can read the pseudo code, and that you can convert it into working programs with a minimum of trouble.

Almost all algorithms have parameters that affect their performance. Science is about quantitative models of our physical world, while art tries to express the qualitative content of our lives. When you read this book you will encounter lots of parameters that are user-defined, together with evasive statements like "pick a value for k that is close to 1", or "don't use high values for m". What do instructions such as these mean? Lots of things: (i) we don't have better advice; (ii) the inventor of the algorithm tried lots of values, and values in the range mentioned produced the best results for her or him; (iii) 0.99 is closer to 1 than 0.95, and 22 is higher than 1.32, you may never know which choice is better, and (unfortunately) this can make all the difference in your application; (iv) sometimes we don't know why things work the way they do, but we should be happy if they work right this time - call it voodoo, or call it luck, but if it works, take it.

Is this cynical? No, it's practical. Science is *NOT* exact, it's a sequence of successively better approximations by models we invent to the physical reality of processes we initiate, observe or control. There's a lot of art in science, and this is nowhere more evident than in pattern recognition, because here, the data always have the last word. We are always at the mercy of an unanticipated situation in the data; unusual structures, missing observations, improbable events that cause outliers, uncertainty about the interactions between variables, useless choices for numerical representation, sensors that don't respect our design goals, computers that lose bits, computer programs that have an undetected flaw, and so on. When you read about and experiment with algorithmic parameters, have an open mind - anything is possible, and usually is.

The data Most of the numerical examples use small data sets that may seem contrived to you, and some of them are. There is much to be said for the pedagogical value of using a few points in the plane when studying and illustrating properties of various models. On the other hand, there are certain risks too. Sometimes conclusions that are legitimate for small, specialized data sets become invalid in the face of large numbers of samples, features and classes. And of course, time and space complexity make their presence felt in very unpredictable ways as problem size grows.

There is another problem with data sets that everyone probably knows about, but that is much harder to detect and document, and that problem goes under the heading of, for example, "*will the real Iris data please stand up*?". Anderson's (1935) Iris data, which we think was first published in Fisher (1936), has become a popular set of labeled data for testing - and especially for comparing - clustering algorithms and classifiers. It is of course entirely appropriate and in the spirit of scientific inquiry to make and publish comparisons of models and their performance on common data sets, and the

pattern recognition community has used Iris in perhaps a thousand papers for just this reason - - - or have we?

During the writing of this book we have discovered - perhaps others have known this for a long time, but we didn't - that there are at least two (and hence, probably half a dozen) different, well publicized versions of Iris. Specifically, vector 90, class 2 (Iris Versicolor) in Iris has the coordinates (5.5, 2.5, 4, 1.3) on p. 566, Johnson and Wichern (1992); and has the coordinates (5.5, 2.5, 5, 1.3) on p. 224 in Chien (1978). YIKES !! For the record, we are using the Iris data as published in Fisher (1936) and repeated in Johnson and Wichern (1992). We will use *Iris (?)* when we are not sure what data were used.

What this means is that many of the papers you have come to know and love that compare the performance of this and that using Iris may in fact have examples of algorithms that were executed using different data sets! What to do? Well, there isn't much we can do about this problem. We have checked our own files, and they all contain the data as listed in Fisher (1936) and Johnson and Wichern (1992). That's not too reassuring, but it's the best we can do. We have tried to check which Iris data set was used in the examples of other authors that are discussed in this book, but this is nearly impossible. We do not guarantee that all the results we discuss for "the" Iris data really pertain to the same numerical inputs. Indeed, the "Lena" image is the Iris data of image processing, - after all, the original Lena was a poor quality, 6 bit image, and more recent copies, including the ones we use in this book, come to us with higher resolution. To be sure, there is only one analog Lena (although PLAYBOY ran many), but there are probably many different digital Lenae.

Data get corrupted many ways, and in the electronic age, it should not surprise us to find (if we can) that this is a fairly common event. Perhaps the best solution to this problem would be to establish a central repository for common data sets. This has been tried several times without much success. Out of curiosity, on September 7, 1998 we fetched Iris from the anonymous FTP site "ftp.ics.uci.edu" under the directory "pub/machine-learning-databases", and discovered not one, but *two* errors in it! Specifically, two vectors in Iris Sestosa were wrong: vector 35 in Fisher (1936) is (4.9, 3.1, 1.5, 0.2) but in the machine learning electronic database it had coordinates (4.9, 3.1, 1.5, 0.1); and vector 38 in Fisher is (4.9, 3.6, 1.4, 0.1), but in the electronic database it was (4.9, 3.1, 1.5, 0.1). Finally, we are aware of several papers that used a version of Iris obtained by multiplying every value by 10, so that the data are integers, and the papers involved discuss 10*Iris as if they thought it was Iris. We don't think there is a way to correct all the databases out there which contain similar mistakes (we trust that the machine learning database will be fixed after our alert), but we have included a listing of Iris in Appendix 2 of this book (and, we hope it's right). What all this means

for you, the pattern recognition aficionado is this: *pattern recognition* **is** *data, and not all data are created equally, much less replicated faithfully!*

Numerical results We have tried to give you all the information you need to *replicate* the outputs we report in numerical examples. There are a few instances where this was not possible (for example, when an iterative procedure was initialized randomly, or when the results were reported in someone's paper 10 or 15 years ago, or when the authors of a paper we discuss simply could not supply us with more details), and of course it's always possible that the code we ran implemented something other than we thought it did, or it simply had undetected programming errors. Also, we have rounded off or truncated the reported results of many calculations to make tables fit into the format of the book. Let us know if you find substantial differences between outputs you get (or got) and the results we report.

The references More than one reference system is one too many. We chose to reference books and papers by last names and years. As with any system, this one has advantages and disadvantages. Our scheme lets you find a paper quickly if you know the last name of the first author, but causes the problem of appending "a", "b" and so on to names that appear more than once in the same year. There may be a mistake or two, or even $O(n)$ of them. Again, please let us know about it. We have divided the references into two groups: those actually cited in the text, and a second set of references that point to related material that, for one reason or another, just didn't find their way into the text discussion. Many of these uncited papers are excellent - please have a look at them.

The acronyms Acronyms, like the plague, seem to spread unchecked through the technical literature of pattern recognition. We four are responsible for quite a few of them, and so, we can hardly hold this bad habit against others. This book has several hundred acronyms in it, and we know you won't remember what many of them mean for more than a few pages. Consequently, Appendix 1 is a tabulation of the acronyms and abbreviations used in the text.

Acknowledgments The authors wish to acknowledge their gratitude to the following agencies, who graciously supplied partial support as shown during the writing of this book:

J. C. Bezdek :	ONR grant # N00014-96-1-0642 NSF grant # IRI-9003252
J. M. Keller :	ONR grant # N00014-96-1-0439 ARO MURI grant # DAAG55-97-1-0014
R. Krishnapuram:	ONR grant # N00014-96-1-0439 NSFgrant # IRI-9800899

We also want to express our thanks to Andrea Baraldi, Alma Blonda, Larry Hall, Lucy Kuncheva and Thomas Runkler, all of whom were kind enough to review various parts of the manuscript and/or supplied us with computations for several examples that we could not find in the literature, and whose helpful comments save us at least a few embarrassments.

The quotes Everyone nowadays seems to have a pithy quote at each chapter head, at the end of each email, on their web page, tattooed on their leg, etc., so we wanted to have some too. Rather than choose one quote for the book that all of us could live with (quite a range of tastes exists amongst us four), we decided to each supply one quote for this preface. We give the quotes here, but don't identify who contributed each one. That will be revealed in the pages of this volume - but only to those readers alert enough to *recognize the patterns*.

"What use are all these high-flying vaunts of yours?
O King of Birds! You will be the world's laughing stock.
What a marvel would it be if the hare
were to void turd the size of elephant dung!"

Vishnu Sharma, in Panchatantra, circa AD 400

"Only the mediocre are always at their best"

Blue Wave, circa 1995

"All uncertainty is fruitful ... so long as it is accompanied by the wish to understand"

Antonio Machado, Juan de Mairena, 1943

"You gotta pay your dues if you want to play the blues, and you know that don't come easy"

Ringo Starr, circa 1973

You may think you know which of us contributed each of these quotes - but you might be surprised. Life is full of surprises, and so is this book. We hope you enjoy both.

Jim Bezdek
Jim Keller
Rags Krishnapuram
Nik Pal

1 Pattern Recognition

1.1 Fuzzy models for pattern recognition

There is no lack of definitions for the term pattern recognition. Here are a few that we like.

Fukunaga (1972, p. 4): "pattern recognition consists of two parts: feature selection and classifier design."

Duda and Hart (1973, p. vii) "pattern recognition, a field concerned with machine recognition of meaningful regularities in noisy or complex environments".

Pavlidis (1977, p. 1): "the word *pattern* is derived from the same root as the word *patron* and, in its original use, means something which is set up as a perfect example to be imitated. Thus pattern recognition means the identification of the ideal which a given object was made after."

Gonzalez and Thomason (1978, p. 1) : "Pattern recognition can be defined as the categorization of input data into identifiable classes via the extraction of significant features or attributes of the data from a background of irrelevant detail."

Bezdek (1981, p. 1) : "pattern recognition is *a search* for *structure* in *data*."

Schalkoff (1992, p. 2) " Pattern recognition (PR) is the science that concerns the description or classification (recognition) of measurements."

And here is our favorite, because it comes from the very nice book by Devijver and Kittler (1982, p. 2), titled *Pattern Recognition: A Statistical Approach*: "pattern recognition is a very broad field of activities with very fuzzy borders" !!!

What all these definitions should tell you is that it's pretty hard to know what to expect from a book with the term pattern recognition in its title. You will find texts that are mostly about computer science topics such as formal language theory and automata design (Fu, 1982), books about statistical decision theory (Fukunaga, 1972, 1991), books about fuzzy mathematics and models (Bezdek, 1981), books about digital hardware (Serrano-Gotarredona et al., 1998), handbooks (Ruspini et al., 1998), pure math books, books that contain only computer programs, books about graphical approaches, and so on. The easiest, and we think, most accurate overall description of this field is to say that it is about feature analysis, clustering, and classifier design, and that is what this book is about - the use of fuzzy models in these three disciplines.

Regardless of how it is defined, there are two major approaches to pattern recognition, numerical and *syntactic*. With the exception of Section 4.10, this book is exclusively concerned with the numerical approach. We characterize numerical pattern recognition with the four major areas shown in Figure 1.1. The nodes in Figure 1.1 are *not* independent. In practice, a successful pattern recognition system is developed by iteratively revisiting the four modules until the system satisfies (or is at least optimized for) a given set of performance requirements and/or economic constraints.

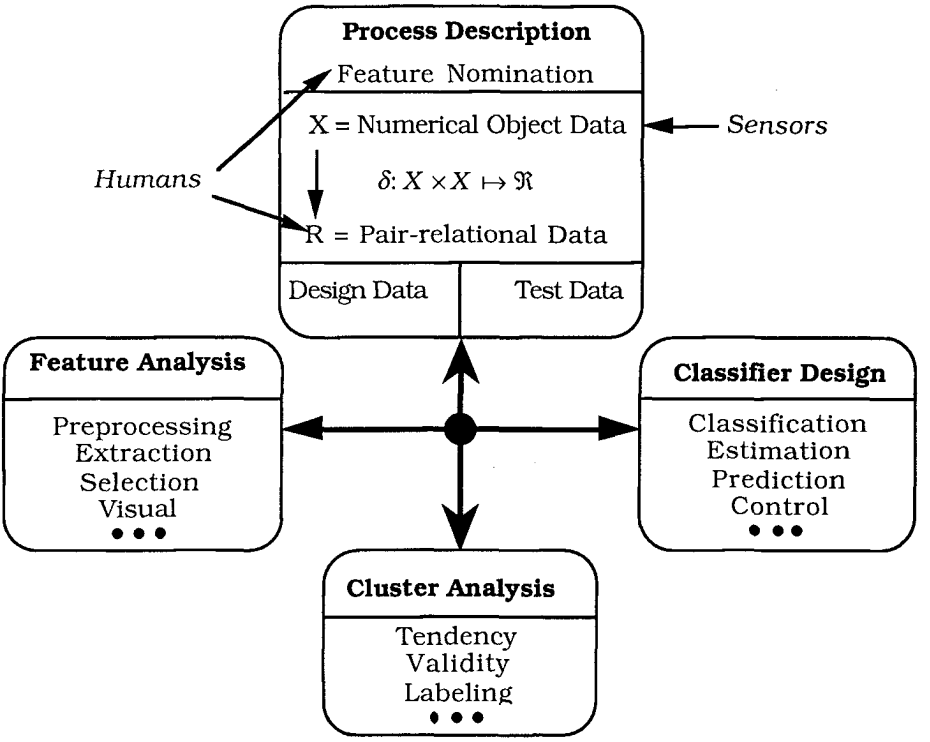

Figure 1.1 Typical elements of numerical pattern recognition

The upper block of Figure 1.1 - *process description* - is always done by humans. Things that must be accomplished here include the selection of a model type, features to be measured and sensors that can collect the data. This important phase of system design is not well represented in the literature because there are many factors such as time, space, weight, cost, speed, etc. that are too problem-dependent to admit much generality. You need to give careful thought to process description because your decisions here will be reflected in the ultimate performance of your system.

Notation Vectors are boldface ($\mathbf{x}$, $\mathbf{v}$, $\mathbf{V}$, etc.); $\mathbf{x} \in \Re^p$ is the $p \times 1$ matrix $\mathbf{x} = (x_1, \ldots, x_p)^T$. Matrices and set names are not shown boldface (even though a $c \times p$ matrix U is a vector in $\Re^{cp} = \Re^c \times \Re^p$). For the matrix $U \in \Re^{cp}$, we may write the i-th row as $\mathbf{U}_{(i)} \in \Re^p$, and the k-th column as $\mathbf{U}_k \in \Re^c$. By this convention, when interpreting U as a $cp \times 1$ column vector, we may write $\mathbf{U} = (\mathbf{U}_1, \ldots, \mathbf{U}_p) = (\mathbf{U}_{(1)}, \ldots, \mathbf{U}_{(p)})^T \in \Re^{cp}$. When interpreting the rows or columns of a matrix as a set, we use set brackets; e.g., the c rows $\mathbf{U} = (\mathbf{U}_{(1)}, \ldots, \mathbf{U}_{(c)}) \in \Re^{cp} \leftrightarrow U = \{\mathbf{U}_{(1)}, \ldots, \mathbf{U}_{(c)}\} \subset \Re^p$. We use $\mathbf{0}$ for the zero vector in all vector spaces; specifically, in both $\Re^p$ and $\Re^{cp}$.

Two data types are used in numerical pattern recognition: *object data* (feature or pattern vectors); and (pairwise) *relational data* (similarities, proximities, etc.). Object data are represented throughout the volume as $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\} \subset \Re^p$, a set of n feature vectors in *feature space* $\Re^p$. Writers in some fields call the features of each object "attributes", and others call them "characteristics". The j-th object is a physical entity such as a tank, medical patient, stock report, etc. *Column vector* $\mathbf{x}_j$ is it's numerical representation; x_{kj} is the k-th *feature* or *attribute value* associated with object j. Features can be either continuously or discretely valued in $\Re$.

We will also deal with non-numerical data called *categorical data* in Chapter 4. Categorical data have no natural order. For example, we can represent animals with numerical attributes such as number of legs, weight, etc. ; or we might describe each one with categorical attributes such as skin texture, which itself has values such as furry, feathery, etc. When needed, we denote the objects themselves as $O = \{o_1, o_2, \ldots, o_n\}$. Chapter 2 is about clustering numerical object data.

Instead of object data, we may have a set of (mn) numerical *relationships*, say $\{r_{jk}\}$, between *pairs* of objects (o_j, o_k) in $O_1 \times O_2$, $|O_1| = m$, $|O_2| = n$. The number r_{jk} represents the extent to which $o_j \in O_1$ is related to $o_k \in O_2$ in the sense of some binary relation ρ. It is convenient to array the relational values as an $m \times n$ *relation matrix* $R = [r_{jk}] = [\rho(o_j, o_k)]$. Many functions can convert object data into relational data. For example, every metric (distance measure) δ on $\Re^p \times \Re^p$ produces a square (dis)-similarity relation matrix $R(X; \delta)$ on the n objects represented by X, as shown in Figure 1.1. If every r_{jk} is in {0, 1}, R is a *crisp* binary relation. If any r_{jk} is in [0, 1], we call R a *fuzzy* binary relation. Chapter 3 is about clustering relational data.

One of the most basic structures in pattern recognition is the *label vector*. No matter what kind of data you have (including the case of n objects as opposed to numerical data that represent them), there are four types of class labels - crisp, fuzzy, probabilistic and possibilistic. Letting n be the number of objects (or feature vectors or number of rows and columns in relational data) integer c denote the number of classes, $1 \le c \le n$. Ordinarily, c will not be 1 or n, but we admit this possibility to handle special cases that sometimes arise. We define three sets of label vectors in $\Re^c$ as follows:

$$N_{pc} = \left\{ \mathbf{y} \in \Re^c : y_i \in [0,\ 1]\ \ \forall\ i,\ y_i > 0\ \ \exists\ i \right\} = [0,1]^c - \{\mathbf{0}\}; \tag{1.1}$$

$$N_{fc} = \left\{ \mathbf{y} \in N_{pc} : \sum_{i=1}^{c} y_i = 1 \right\} \quad ; \tag{1.2}$$

$$N_{hc} = \left\{ \mathbf{y} \in N_{fc} : y_i \in \{0,1\}\ \forall\, i \right\} = \left\{ \mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_c \right\} \quad . \tag{1.3}$$

In (1.1) $\mathbf{0}$ is the *zero vector* in $\Re^c$. Note that $N_{hc} \subset N_{fc} \subset N_{pc}$. Figure 1.2 depicts these sets for c = 3. N_{hc} is the canonical (unit vector) basis of Euclidean c-space, so $\mathbf{e}_i = (0,\ 0\ ,\ldots,\underbrace{1}_{i},\ldots,\ 0)^T$, the i-th vertex of N_{hc}, is the *crisp label* for class i, $1 \le i \le c$.

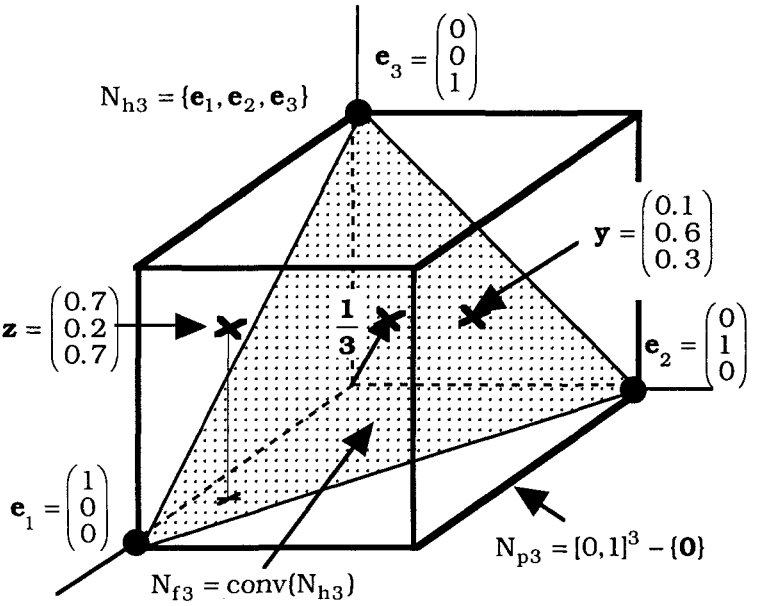

Figure 1.2 Label vectors for c = 3 classes

The set N_{fc}, a piece of a hyperplane, is the convex hull of N_{hc}. The vector $\mathbf{y} = (0.1, 0.6, 0.3)^T$ is a constrained label vector; its entries lie between 0 and 1, and sum to 1. The *centroid* of N_{fc} is the equimembership vector $\mathbf{1/c} = (1/c, \ldots, 1/c)^T$. If $\mathbf{y}$ is a label vector for some $\mathbf{x} \in \Re^p$ generated by, say, the fuzzy c-means clustering method, we call $\mathbf{y}$ a *fuzzy label* for $\mathbf{x}$. If $\mathbf{y}$ came from a method such as maximum likelihood estimation in mixture decomposition, $\mathbf{y}$ would be a *probabilistic label*. In this case, $\mathbf{1/c}$ is the unique point of equal probabilities for all c classes.

$N_{pc} = [0, 1]^c - \{\mathbf{0}\}$ is the unit hypercube in $\Re^c$, *excluding the origin*. Vectors such as $\mathbf{z} = (0.7, 0.2, 0.7)^T$ with each entry between 0 and 1 that are otherwise unrestricted are *possibilistic labels* in N_{p3}. Possibilistic labels are produced by possibilistic clustering algorithms (Krishnapuram and Keller, 1993) and by computational neural networks that have unipolar sigmoidal transfer functions at each of c output nodes (Zurada, 1992).

Most pattern recognition models are based on finding statistical or geometrical properties of substructures in the data. Two of the key concepts for describing geometry are angle and distance. Let A be any positive-definite p×p matrix. For vectors $\mathbf{x}, \mathbf{v} \in \Re^p$, the functions $\langle\ \rangle_A : \Re^p \times \Re^p \mapsto \Re$, $\|\ \|_A : \Re^p \mapsto \Re^+$, and $\delta_A : \Re^p \times \Re^p \mapsto \Re^+$

$$\langle \mathbf{x}, \mathbf{v} \rangle_A = \mathbf{x}^T A \mathbf{v} \quad ; \tag{1.4}$$

$$\|\mathbf{x}\|_A = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle_A} = \sqrt{\mathbf{x}^T A \mathbf{x}} \quad ; \text{ and} \tag{1.5}$$

$$\delta_A(\mathbf{x}, \mathbf{v}) = \|\mathbf{x} - \mathbf{v}\|_A = \sqrt{(\mathbf{x} - \mathbf{v})^T A (\mathbf{x} - \mathbf{v})} \quad , \tag{1.6}$$

are the inner product (dot product, scalar product), *norm* (length), and *norm metric* (distance) induced on $\Re^p$ by weight matrix A. We say that $\mathbf{x}$ and $\mathbf{v}$ are *orthogonal* (normal, perpendicular) if their dot product is zero, $\langle \mathbf{x}, \mathbf{v} \rangle_A = \mathbf{x}^T A \mathbf{v} = 0$. Sometimes we write $\mathbf{x} \perp_A \mathbf{v}$ to indicate this, and note particularly that orthogonality is always *relative to* matrix A that induces the inner product.

Equation (1.6) defines an infinite family of inner product induced distances, the most important three of which, together with their common names and inducing matrices, are:

$$\|\mathbf{x} - \mathbf{v}\|_I = \sqrt{(\mathbf{x} - \mathbf{v})^T (\mathbf{x} - \mathbf{v})} \qquad \textit{Euclidean}, A = I_p \quad ; \tag{1.7}$$

$$\|\mathbf{x}-\mathbf{v}\|_{D^{-1}} = \sqrt{(\mathbf{x}-\mathbf{v})^T D^{-1}(\mathbf{x}-\mathbf{v})} \qquad \textit{Diagonal, } A=D^{-1} \quad ; \quad (1.8)$$

$$\|\mathbf{x}-\mathbf{v}\|_{M^{-1}} = \sqrt{(\mathbf{x}-\mathbf{v})^T M^{-1}(\mathbf{x}-\mathbf{v})} \qquad \textit{Mahalanobis, } A=M^{-1}. \quad (1.9)$$

Notation In (1.7) I_p is the $p \times p$ *identity matrix*. Henceforth, we drop the subscript I_p, writing the Euclidean forms of (1.4)-(1.6) more simply as $\langle \mathbf{x}, \mathbf{v} \rangle$, $\|\mathbf{x}\|$ and $\|\mathbf{x}-\mathbf{v}\|$ respectively.

Equations (1.8) and (1.9) use $M = \mathrm{cov}(X) = \sum_{k=1}^{n} (\mathbf{x}_k - \bar{\mathbf{v}})(\mathbf{x}_k - \bar{\mathbf{v}})^T / n$, the *covariance matrix* of X, and $\bar{\mathbf{v}} = \sum_{k=1}^{n} \mathbf{x}_k / n$, the *grand mean* of X. We will always indicate sample means as in statistics, with an overbar. The matrix D is the diagonal matrix extracted from M by deletion of its off-diagonal entries, D = diag(M). D is *not* the diagonalized form of M.

A second infinite family of lengths and distances that are commonly used in pattern recognition are the *Minkowski norm* and *Minkowski norm metrics*

$$\|\mathbf{x}\|_q = \left(\sum_{j=1}^{p} |x_j|^q \right)^{\frac{1}{q}}, \quad q \geq 1 \qquad ; \qquad (1.10)$$

$$\delta_q(\mathbf{x}, \mathbf{v}) = \|\mathbf{x}-\mathbf{v}\|_q = \left(\sum_{j=1}^{p} |x_j - v_j|^q \right)^{\frac{1}{q}}, \quad q \geq 1 \qquad . \qquad (1.11)$$

Only three Minkowski distances are commonly used in pattern recognition, and the Minkowski 2-norm is just the Euclidean norm, $\|\mathbf{x}-\mathbf{v}\|_2 = \|\mathbf{x}-\mathbf{v}\|$:

$$\|\mathbf{x}-\mathbf{v}\|_1 = \left(\sum_{j=1}^{p} |x_j - v_j| \right) \qquad \text{City Block (1-norm); } q=1; \qquad (1.12)$$

$$\|\mathbf{x}-\mathbf{v}\|_2 = \left(\sum_{j=1}^{p} |x_j - v_j|^2 \right)^{\frac{1}{2}} \qquad \text{Euclidean (2-norm); } q=2; \qquad (1.13)$$

$$\|\mathbf{x}-\mathbf{v}\|_\infty = \underbrace{\max}_{1 \leq j \leq p} \{ |x_j - v_j| \} \qquad \text{Sup or Max norm; } q \to \infty. \qquad (1.14)$$

A *classifier* is any function $\mathbf{D}: \Re^p \mapsto N_{pc}$. The value $\mathbf{y} = \mathbf{D}(\mathbf{z})$ is the label vector for $\mathbf{z}$ in $\Re^p$. $\mathbf{D}$ is a *crisp classifier* if $\mathbf{D}[\Re^p] = N_{hc}$; otherwise, the classifier is fuzzy or probabilistic or possibilistic. Designing a classifier simply means finding the parameters of a "good" $\mathbf{D}$. This can be done with data, or it might be done by an expert without data. If the data are labeled, finding $\mathbf{D}$ is called *supervised learning*; otherwise, the problem is *unsupervised learning*. Notice that we use the terms supervised and unsupervised to specifically connote the use of labeled or unlabeled data - it is the *labels* that do (or do not) supervise the design. When an expert designs a classifier, this is certainly supervised design, but in a much broader sense than we mean here. Chapter 4 is about fuzzy models for classifier design.

Since definite class assignments are usually the ultimate goal of classification and clustering, outputs of algorithms that produce label vectors in N_{pc} or N_{fc} are usually transformed into crisp labels. Most non-crisp classifiers are converted to crisp ones using the function $\mathbf{H}: N_{pc} \mapsto N_{hc}$,

$$\mathbf{H}(\mathbf{y}) = \mathbf{e}_i \Leftrightarrow \left\|\mathbf{y} - \mathbf{e}_i\right\| < \left\|\mathbf{y} - \mathbf{e}_j\right\| \Leftrightarrow y_i > y_j \quad ; \quad j \neq i \quad . \tag{1.15}$$

In (1.15) ties are resolved arbitrarily. $\mathbf{H}$ finds the crisp label vector $\mathbf{e}_i$ in N_c closest (in the Euclidean sense) to $\mathbf{y}$. Alternatively, $\mathbf{H}$ finds the index of the *maximum coordinate* of $\mathbf{y}$, and assigns the corresponding crisp label to the object vector, say $\mathbf{z}$, that $\mathbf{y}$ labels. The rationale for using $\mathbf{H}$ depends on the algorithm that produces $\mathbf{y}$. For example, using (1.15) for outputs from the k-nearest neighbor rule is simple majority voting. If $\mathbf{y}$ is obtained from mixture decomposition, using $\mathbf{H}$ is Bayes decision rule - label $\mathbf{z}$ by its class of maximum posterior probability. And if the labels are fuzzy, this is called defuzzification by the maximum membership rule. We call the use of $\mathbf{H}$ *hardening*.

1.2 Why fuzzy pattern recognition?

Rather than conclude the volume with the information in this subsection, it is provided here to answer a basic question you might have at this point: *should you read on*? Retrieval from the *Science Citation Index* for years 1994-1997 on titles and abstracts that contain the keyword combinations "fuzzy" + either "clustering" or "classification" yielded 460 papers. Retrievals against "fuzzy" + either "feature selection" or "feature extraction" yielded 21 papers. This illustrates that the literature contains a large body of work on fuzzy clustering and classifier design, and relatively fewer studies of fuzzy models for feature analysis. Work in this last area is widely

scattered because feature analysis is very data and problem-dependent, and hence, is almost always done on a case by case basis.

A more interesting metric for the importance of fuzzy models in pattern recognition lies in the diversity of applications areas represented by the titles retrieved. Here is a partial sketch:

Chemistry: analytical, computational, industrial, chromatography, food engineering, brewing science.

Electrical Engineering: image and signal processing, neural networks, control systems, informatics, automatics, automation, robotics, remote sensing and control, optical engineering, computer vision, parallel computing, networking, instrumentation and measurement, dielectrics, speech recognition, solid state circuits.

Geology/Geography: photogrammetry, geophysical research, geochemistry, biogeography, archeology.

Medicine: magnetic resonance imaging, medical diagnosis, tomography, roentgenology, neurology, pharmacology, medical physics, nutrition, dietetic sciences, anesthesia, ultramicroscopy, biomedicine, protein science, neuroimaging, drug interaction.

Physics: astronomy, applied optics, earth physics.

Environmental Sciences: soil sciences, forest and air pollution, meteorology, water resources.

Thus, it seems fair to assert that this branch of science and engineering has established a niche as a useful way to approach pattern recognition problems. The rest of this volume is devoted to some of the basic models and algorithms that comprise fuzzy numerical pattern recognition.

1.3 Overview of the volume

Chapter 2 discusses clustering with objective function models using object data. This chapter is anchored by the crisp, fuzzy and possibilistic c-means models and algorithms to optimize them that are discussed in Section 2.2. There are many generalizations and relatives of these three families. We discuss relatives and generalizations of the c-means models for both volumetric (cloud shaped) and shell clusters in Section 2.3. Roughly speaking, these two cases can be categorized as point and non-point prototype models. Section 2.3 also contains a short subsection on recent developments in the new area of robust clustering. Chapter 2 contains a long section on methods for validation of clusters after they are found - the important and very difficult problem of cluster validity. Separate subsections discuss methods that attempt to

validate volumetric and shell type clusters; and this section concludes with a discussion of fuzzy versions of several well known statistical indices of validity. This is followed by a short section on feature analysis with references to a very few fuzzy methods for problems in this domain. Finally, we close Chapter 2 (and all subsequent chapters as well) with a section that contains comments and related references for further reading.

Chapter 3 is about two types of relational clustering: methods that use decompositions of relation matrices; and methods that rely on optimization of an objective function of the relational data. This is a much smaller field than clustering with objective function methods. The main reason that relational models and algorithms are less well developed than those for object data is that sensors in fielded systems almost always collect object data. There are, however, some very interesting applications that depend on relational clustering; for example, data mining and information retrieval in very large databases. We present the main topics of this area in roughly the same chronological order as they were developed. Applications of relational clustering are also discussed in the handbook volume devoted to information retrieval.

Chapter 4 discusses fuzzy models that use object data for classifier design. Following definitions and examples of the nearest single and multiple prototype classifiers, we discuss several sequential methods of prototype generation that were not covered in Chapter 2. Next, k-nearest neighbor rule classifiers are presented, beginning with the classical crisp k-nearest neighbor rule, and continuing through both fuzzy and possibilistic generalizations of it. Another central idea covered in Chapter 4 is the use of the fuzzy integral for data fusion and decision making in the classification domain. Following this, rule based designs are introduced through crisp and fuzzy decision trees in Section 4.6, which contains material about the extraction of fuzzy rules for approximation of functions from numerical data with clustering.

Chapter 4 next presents models and algorithms that draw their inspiration from *neural-like networks* (NNs). Two chapters in Nguyen and Sugeno (1998) by Pedrycz et al.,(1998) and Prasad (1998) discuss the use of fuzzy neurons and fuzzy NNs in the context of control and functional approximation. These chapters provide good ancillary reading to our presentation of related topics in the context of pattern recognition. The *feed forward* multilayered perceptron trained by *back propagation* (FFBP) is the dominant structure underlying "fuzzy neural networks" (neurofuzzy computing, etc.), so our discussion begins with this network as the standard classifier network. Then we present some generalizations of the standard node functions that are sometimes called fuzzy neurons. We discuss and illustrate perceptrons, multilayered perceptrons, and aggregation networks for classification. Then we discuss the crisp

and several fuzzy generalizations of *adaptive resonance theory* (ART), including a short subsection on radial basis function networks. Section 4.9 is concerned with the increasingly important topic of classifier fusion (or multistage classification). The last section in Chapter 4 is a short section on the use of fuzzy models in syntactic pattern recognition. Our Chapter 4 comments include some material on feature analysis in the context of classifier design.

Chapter 5 is about image processing and computer vision. It is here that the models and algorithms discussed in previous chapters find realizations in an important application domain. Chapter 5 begins with low level vision approaches to image enhancement. Then we discuss edge detection and edge following algorithms. Several approaches to the important topic of image segmentation are presented next, followed by boundary description and surface approximation models. The representation of image objects as fuzzy regions is followed by a section on spatial relations. The last section in Chapter 5 discusses high level vision using fuzzy models. Chapter 7.3.2 of volume 7 of this handbook (Bezdek and Sutton, 1998) contains an extended discussion of fuzzy models for image processing in medical applications.

1.4 Comments and bibliography

There are many good treatments of deterministic, statistical and heuristic approaches to numerical pattern recognition, including the texts of Duda and Hart (1973), Tou and Gonzalez (1974), Devijver and Kittler (1982), Pao (1989) and Fukunaga (1991). Approaches based on neural-like network models are nicely covered in the texts by Zurada (1992) and Haykin (1994).

The earliest reference to the use of fuzzy sets in numerical pattern recognition was Bellman, Kalaba and Zadeh (1966). RAND Memo RM-4307-PR, October, 1964, by the same authors had the same title, and was written before Zadeh (1965). Thus, the first *application* envisioned for fuzzy models seems to have been in pattern recognition.

Fuzzy techniques for numerical pattern recognition are now fairly mature. Good references include the texts by Bezdek (1981), Kandel (1982), Pal and Dutta-Majumder (1986) and the edited collection of 51 papers by Bezdek and Pal (1992). Chi et al. (1997) is the latest entrant into this market, with a title so close to ours that it makes you wonder how many of these entries the market will bear. Surveys of fuzzy models in numerical pattern recognition include Keller and Qiu(1988), Pedrycz (1990b), Pal (1991), Bezdek (1993), Keller and Krishnapuram (1994), Keller et al. (1994) and Bezdek et al. (1997a).

2 Cluster Analysis for Object Data

2.1 Cluster analysis

Figure 2.1 portrays cluster analysis. This field comprises three problems: tendency assessment, clustering and validation. Given an unlabeled data set, ① *is there* substructure in the data? This is *clustering tendency* - should you look for clusters at all? Very few methods - fuzzy or otherwise - address this problem. Panayirci and Dubes (1983), Smith and Jain (1984), Jain and Dubes (1988), Tukey (1977) and Everitt (1978) discuss statistical and informal graphical methods (visual displays) for deciding what - if any - substructure is in unlabeled data.

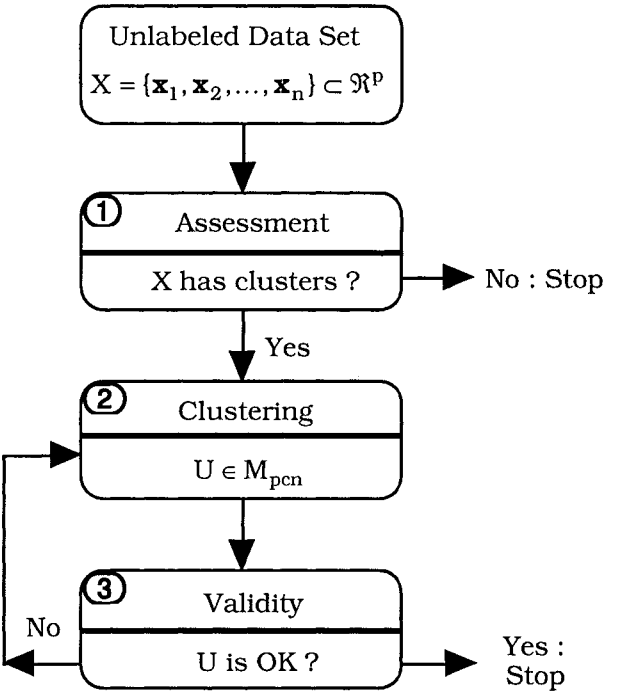

Figure 2.1 Cluster analysis: three problems

Once you decide to look for clusters (called U in ②, Figure 2.1), you need to choose a model whose measure of mathematical similarity may capture structure in the sense that a human might perceive it. This question - what criterion of similarity to use? - lies at the heart of all clustering models. We will be careful to distinguish between a model, and methods (algorithms) used to solve or optimize it. There are objective function (global criteria) and graph-theoretic (local criteria) techniques for both relational and object data.

Different algorithms produce different partitions of the data, and it is never clear which one(s) may be most useful. Once clusters are obtained, how shall we pick the best clustering solution (or solutions)? Problem ③ in Figure 2.1 is *cluster validity*, discussed in Section 2.4.

Problem ② in Figure 2.1 is *clustering* (or unsupervised learning) in unlabeled data set $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, which is the assignment of (hard *or* fuzzy *or* probabilistic *or* possibilistic) label vectors to the $\{\mathbf{x}_k\}$. The word *learning* refers to learning good labels (and possibly prototypes) for the clusters in the data.

A *c-partition* of X is a c×n matrix $U = [\mathbf{U}_1\ \mathbf{U}_k \ldots \mathbf{U}_n] = [u_{ik}]$, where $\mathbf{U}_k$ denotes the k-th column of U. There are three sets of c-partitions whose columns correspond to the three types of label vectors discussed in Chapter 1

$$M_{pcn} = \left\{ U \in \Re^{cn} : \mathbf{U}_k \in N_{pc}\, \forall k; 0 < \sum_{k=1}^{n} u_{ik}\ \forall i \right\} \quad ; \tag{2.1}$$

$$M_{fcn} = \left\{ U \in M_{pcn} : \mathbf{U}_k \in N_{fc}\ \forall k \right\} \quad ; \tag{2.2}$$

$$M_{hcn} = \left\{ U \in M_{fcn} : \mathbf{U}_k \in N_{hc} \forall k \right\} \quad . \tag{2.3}$$

Equations (2.1), (2.2) and (2.3) define, respectively, the sets of possibilistic, fuzzy or probabilistic, and crisp c-partitions of X. Each column of U in M_{pcn} (M_{fcn}, M_{hcn}) is a label vector from N_{pc} (N_{fc}, N_{hc}). Note that $M_{hcn} \subset M_{fcn} \subset M_{pcn}$. Our notation is chosen to help you remember these structures; M = (membership) matrix, h=crisp (hard), f=fuzzy (or probabilistic), p=possibilistic, c=number of classes and n=number of data points in X.

Notation For U in M_{fcn} c=1 is represented uniquely by the hard 1-partition $\mathbf{1}_n = [\underbrace{1\ \ 1\ \ \cdots\ \ 1}_{n\ \text{times}}]$, which asserts that all n objects belong to a single cluster; and c=n is represented uniquely by $U = I_n$, the n×n identity matrix, up to a permutation of columns. In this case each object is in its own singleton cluster. Crisp partitions have a familiar set-theoretic description that is equivalent to (2.1). When $U = \{X_1, \ldots, X_c\}$ is a crisp c-partition, the c crisp subsets $\{X_i\} \subset X$ satisfy $\bigcup_i X_i = X$; $X_i \cap X_j = \varnothing$ if $i \neq j$; and $X_i \neq \varnothing\ \forall i$. We denote the *cardinality* of a crisp set of n elements as $|X| = n$, and $|X_i| = n_i\ \forall i$.

Choosing c=1 or c=n rejects the hypothesis that X contains clusters in these cases. The lack of a column sum constraint for $U \in (M_{pcn} - M_{fcn})$ means that there are infinitely many U's in both $(M_{p1n} - M_{f1n})$ and $(M_{pnn} - M_{fnn})$.

The constraint $0 < \sum_{k=1}^{n} u_{ik} \; \forall i$ in equation (2.1) guarantees that each row in a c-partition contains at least one non-zero entry, so the corresponding cluster is not empty. Relaxing this constraint results in enlarging M_{pcn} to include matrices that have zero rows (empty clusters). From a practical viewpoint this is not desirable, but we often need this superset of M_{pcn} for theoretical reasons. We designate the sets of *degenerate* (crisp, fuzzy, possibilistic) *c-partitions* of X as $(M_{hcn0}, M_{fcn0}, M_{pcn0})$.

The reason these matrices are called *partitions* follows from the interpretation of their entries. If U is crisp or fuzzy, u_{ik} is taken as the *membership* of $\mathbf{x}_k$ in the i-th partitioning fuzzy subset (cluster) of X. If U is probabilistic, u_{ik} is usually the (posterior) probability $p(i \mid \mathbf{x}_k)$ that, given $\mathbf{x}_k$, it came from class (cluster) i. We indicate the statistical context by replacing $U = [u_{ik}]$ with $P = [p_{ik}] = [p(i \mid \mathbf{x}_k)]$. When U is possibilistic, u_{ik} is taken as the possibility that $\mathbf{x}_k$ belongs to class (cluster) i.

Clustering algorithms produce *sets* of label vectors. For fuzzy partitions, the usual method of defuzzification is the application of (1.15) to each column $\mathbf{U}_k$ of matrix U, producing the maximum membership matrix we sometimes call U_{MM} from U. We will formalize this operation as equation (2.10).

Example 2.1 Let $X = \{x_1 = \text{peach}, x_2 = \text{plum}, x_3 = \text{nectarine}\}$, and let c=2. Typical 2-partitions of these three objects are:

	$U_1 \in M_{h23}$			$U_2 \in M_{f23}$			$U_3 \in M_{p23}$		
Object	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$
Peaches	1.0	0.0	0.0	1.0	0.2	0.4	1.0	0.2	0.5
Plums	0.0	1.0	1.0	0.0	0.8	0.6	0.0	0.8	0.6

The nectarine, x_3, is labeled by the last column of each partition, and in the crisp case, it must be (erroneously) given full membership in one of the two crisp subsets partitioning this data. In U_1, x_3 is labeled "plum". Non-crisp partitions enable models to (sometimes!)

avoid such mistakes. The last column of U_2 allocates most (0.6) of the membership of x_3 to the plums class; but also assigns a lesser membership (0.4) to x_3 as a peach. U_3 illustrates possibilistic label assignments for the objects in each class.

Finally, observe that hardening each column of U_2 and U_3 with (1.15) in this example makes them identical to U_1. Crisp partitions of data do not possess the information content to suggest fine details of infrastructure such as hybridization or mixing that are available in U_2 and U_3. Consequently, extract information of this kind before you harden U!

Columns like the ones for the nectarine in U_2 and U_3 serve a useful purpose - lack of strong membership in a single class is a signal to "take a second look". In this example the nectarine is a peach-plum *hybrid*, and the memberships shown for it in the last column of either U_2 or U_3 seem more plausible *physically* than crisp assignment of $\mathbf{x}_3$ to an incorrect class. M_{pcn} and M_{fcn} can be more realistic than M_{hcn} because boundaries between many classes of real objects are badly delineated (i.e., really fuzzy). M_{fcn} reflects the degrees to which the classes share $\{\mathbf{x}_k\}$, because of the constraint inherited from each fuzzy label vector (equation (1.2)) we have $\sum_{i=1}^{c} u_{ik} = 1$. M_{pcn} reflects the degrees of typicality of $\{\mathbf{x}_k\}$ with respect to the prototypical (ideal) members of the classes.

We believe that Bill Wee wrote the first Ph.D. thesis about fuzzy pattern recognition (Wee, 1967); his work is summarized in Wee and Fu (1969). Ruspini (1969) defined M_{fcn}, and Ruspini (1970) discussed the first fuzzy clustering method that produced constrained c-partitions of unlabeled (relational) data. Gitman and Levine (1970) first attempted to decompose "mixtures" (data with multimodality) using fuzzy sets. Other early work includes Woodbury and Clive (1974), who combined fuzziness and probability in a hybrid clustering model. In the same year, Dunn (1974a) and Bezdek (1974a) published papers on the fuzzy c-means clustering model. Texts that contain good accounts of various clustering algorithms include Duda and Hart (1973), Hartigan (1975), Jain and Dubes (1988), Kaufman and Rouseeuw (1990), Miyamoto (1990), Johnson and Wichern (1992), and the most recent members of the fold, Chi et al. (1996a) and Sato et al. (1997).

2.2 Batch point-prototype clustering models

Clustering models and algorithms that optimize them always deliver a c-partition U of X. Many clustering models estimate other

parameters too. The most common parameters besides U that are associated with clustering are sets of vectors we shall denote by $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_c\} \subset \Re^p$. The vector $\mathbf{v}_i$ is interpreted as a *point prototype* (centroid, cluster center, signature, exemplar, template, codevector) for the points associated with cluster i. Point prototypes are regarded as compact representations of cluster structure.

As just defined, $\mathbf{v}_i$ is a point in $\Re^p$, hence a *point-prototype*. Extensions of this idea to prototypes that are not just points in the feature space include $\mathbf{v}_i$'s that are linear varieties, hyperspherical shells, and regression models. General prototype models are covered in Section 2.3. Probabilistic clustering with normal mixtures produces simultaneous estimates of a $c \times n$ partition P (posterior probabilities), c mean vectors $\mathbf{M} = \{\mathbf{m}_1, \ldots, \mathbf{m}_c\}$, c covariance matrices $\{S_1, \ldots, S_c\}$ and c prior probabilities $\mathbf{p} = (p_1, \ldots, p_c)^T$. Some writers regard the triple $(p_i, \mathbf{m}_i, S_i)$ as the prototype for class i; more typically, however, $\mathbf{m}_i$ is considered the point prototype for class i, and other parameters such as p_i and S_i are associated with it through the model.

The basic form of iterative point prototype clustering algorithms in the variables (U, **V**) is

$$(U_t, \mathbf{V}_t) = \mathcal{C}(X: U_{t-1}, \mathbf{V}_{t-1}), t > 0 \quad , \tag{2.4a}$$

where $\mathcal{C}$ stands for the clustering algorithm and t is the index of iteration or recursion. Non-iterative models are dealt with on a case by case basis. When $\mathcal{C}$ is based on optimization of an objective function and *joint* optimization in (U, **V**) is possible, conditions (2.4a) can be written as $(U_t, \mathbf{V}_t) = \mathcal{C}(X: \mathcal{H}_{\mathcal{C}}(U_{t-1}, \mathbf{V}_{t-1}))$, where $\mathcal{H}_{\mathcal{C}}$ is determined by some optimality criterion for the clustering model. More typically however, alternating *optimization* (AO) is used, which takes the form of coupled equations such as

$$U_t = \mathcal{F}_{\mathcal{C}}(\mathbf{V}_{t-1}) \; ; \; \mathbf{V}_t = \mathcal{G}_{\mathcal{C}}(U_t) \text{ [V-initialization]; or} \tag{2.4b}$$

$$\mathbf{V}_t = \mathcal{G}_{\mathcal{C}}(U_{t-1}); U_t = \mathcal{F}_{\mathcal{C}}(\mathbf{V}_t) \text{ [U-initialization].} \tag{2.4c}$$

The iterate sequences in (2.4b) or (2.4c) are equivalent. Both are exhibited to point out that you can start (initialize) and end (terminate) iteration with either U or **V**. Specific implementations use one or the other, and properties of either sequence (such as convergence) automatically follow for iteration started at the opposite set of variables. Examples of clustering models that have (U, **V**) as joint parameters are the batch hard, fuzzy and possibilistic c-means models. Alternating optimization of these models stems from functions $\mathcal{F}_{\mathcal{C}}$ and $\mathcal{G}_{\mathcal{C}}$ which arise from first order necessary

conditions for minimization of the appropriate c-means objective function.

A. The c-means models

The c-means (or k-means) families are the best known and most well developed families of batch clustering models. Why? Probably because they are *least squares* models. The history of this methodology is long and important in applied mathematics because least-squares models have many favorable mathematical properties. (Bell (1966, p. 259) credits Gauss with the invention of the method of least squares for parameter estimation in 1802, but states that Legendre apparently published the first formal exposition of it in 1806.) The optimization problem that defines the *hard* (H), *fuzzy* (F) and *possibilistic* (P) *c-means* (HCM, FCM and PCM, respectively) models is:

$$\underbrace{\min}_{(U,\,\mathbf{V})}\left\{ J_m(U,\mathbf{V};\mathbf{w}) = \sum_{i=1}^{c}\sum_{k=1}^{n} u_{ik}^m D_{ik}^2 + \sum_{i=1}^{c} w_i \sum_{k=1}^{n} (1-u_{ik})^m \right\}, \text{ where} \qquad (2.5)$$

$U \in M_{hcn}$, M_{fcn} or M_{pcn} for HCM, FCM or PCM respectively ;

$\mathbf{V} = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_c) \in \Re^{cp}$; $\mathbf{v}_i \in \Re^p$ is the i-th point prototype ;

$\mathbf{w} = (w_1, w_2, \ldots, w_c)^T$; $w_i \in \Re^+$ is the i-th penalty term (PCM) ;

$m \geq 1$ is the degree of fuzzification ;

$D_{ik}^2 = \|\mathbf{x}_k - \mathbf{v}_i\|_A^2$.

Note especially that $\mathbf{w}$ in (2.5) is a fixed, user-specified vector of positive weights; it is *not* part of the variable set in minimization problem (2.5).

Caveat: Model optima versus human expectations. The presumption in (2.5) is that "good" solutions for a pattern recognition problem - here clustering - correspond to "good" solutions of a mathematical optimization problem chosen to represent the physical process. Readers are warned not to expect too much from their models. In (2.5) the implicit assumption is that pairs (U, $\mathbf{V}$) that are at least local minima for J_m will provide (i) good clusters U, and (ii) good prototypes $\mathbf{V}$ to represent those clusters. What's wrong with this? Well, it's easy to construct a simple data set upon which the *global* minimum of J_m leads to algorithmically suggested substructure that humans will disagree with (example 2.3). The problem? Mathematical models have a very rigid, well-defined idea of what best means, and it is often quite different than that held by human evaluators. There may not be any relationship between clusters that humans regard as "good" and the various types of