
Algorithms for Clustering Data

Anil K. Jain
Richard C. Dubes

Michigan State University



Prentice Hall
Englewood Cliffs, New Jersey 07632

QA278
.J34
1988

Library of Congress Cataloging-in-Publication Data

JAIN, ANIL K., 1948-
Algorithms for clustering data / Anil K. Jain, Richard C. Dubes.
p. cm. — (Prentice Hall advanced reference series)
Bibliography: p.
Includes index.
ISBN 0-13-022278-X
1. Cluster analysis—Data processing. 2. Algorithms. I. Dubes,
Richard C. II. Title.
QA278.J35 1988
519.5'3—dc19

Editorial/production supervision
and interior design: BARBARA MARTINE
Cover design: KAREN STEPHENS
Manufacturing buyer: LORRAINE FUMOSO

Prentice Hall Advanced Reference Series



© 1988 by Prentice-Hall, Inc.
A Division of Simon & Schuster
Englewood Cliffs, New Jersey 07632

The publisher offers discounts on this book when ordered
in bulk quantities. For more information, write:

Special Sales/College Marketing
College Technical and Reference Division
Prentice Hall
Englewood Cliffs, New Jersey 07632

All rights reserved. No part of this book may be
reproduced, in any form or by any means,
without permission in writing from the publisher.

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-022278-X

PRENTICE-HALL INTERNATIONAL (UK) LIMITED, *London*
PRENTICE-HALL OF AUSTRALIA PTY. LIMITED, *Sydney*
PRENTICE-HALL CANADA INC., *Toronto*
PRENTICE-HALL HISPANOAMERICANA, S.A., *Mexico*
PRENTICE-HALL OF INDIA PRIVATE LIMITED, *New Delhi*
PRENTICE-HALL OF JAPAN, INC., *Tokyo*
SIMON & SCHUSTER ASIA PTE. LTD., *Singapore*
EDITORIA PRENTICE-HALL DO BRASIL, LTDA., *Rio de Janeiro*

6-20-90 al

TO NANDITA AND MARYLYN

PREFACE

1 INTRODUCTION

2 DATA REPRESENTATION

- 2.1 Data Types and Data Objects 1
- 2.2 Priority Values 14
- 2.3 Hyper-objects 123
- 2.4 Typed Propositions 125
- 2.5 Hyperobject Propositions 127
- 2.6 Infinite Hyperobjects 129
- 2.7 Hyperobject Propositions 131
- 2.8 Summary 131

3 FINDING THE METHOD AND ALGORITHMS

- 3.1 Finding the Method 132
- 3.2 Finding the Algorithm 133

LINDA HALL LIBRARY
KANSAS CITY, MO.

3

Clustering Methods and Algorithms

Cluster analysis is the process of classifying objects into subsets that have meaning in the context of a particular problem. The objects are thereby organized into an efficient representation that characterizes the population being sampled. In this chapter we present the clustering methods themselves and explain algorithms for performing cluster analysis. Section 3.1 lists the factors involved in classifying objects, and in Sections 3.2 and 3.3 we explain the two most common types of classification. Computer software for cluster analysis is described in Section 3.4. Section 3.5 outlines a methodology for using clustering algorithms to the best advantage. This chapter focuses on the act of clustering itself by concentrating on the inputs to and outputs from clustering algorithms. The need for the formal validation methods in Chapter 4 will become apparent during the discussion.

3.1 GENERAL INTRODUCTION

A clustering is a type of classification imposed on a finite set of objects. As explained in Section 2.2, the relationship between objects is represented in a proximity matrix in which rows and columns correspond to objects. If the objects are characterized as patterns, or points in a d -dimensional metric space, the proximities can be distance between pairs of points, such as Euclidean distance. Unless a meaningful measure of distance, or proximity, between pairs of objects has been established, no meaningful cluster analysis is possible. The proximity matrix is the one and only input to a clustering algorithm.

Clustering is a special kind of classification. See Kendall (1966) for discussion on the relationship between classification and clustering. Figure 3.1 shows a tree of classification problems as suggested by Lance and Williams (1967). Each leaf in the tree in Figure 3.1 defines a different genus of classification problem. The nodes in the tree of Figure 3.1 are defined below.

a. Exclusive versus nonexclusive. An exclusive classification is a partition of the set of objects. Each object belongs to exactly one subset, or cluster. Nonexclusive, or overlapping, classification can assign an object to several classes. For example, a grouping of people by age or sex is exclusive, whereas a grouping by disease category is nonexclusive because a person can have several diseases simultaneously. Shepard and Arabie (1979) provide a review of nonexclusive or overlapping clustering methods. This chapter treats only exclusive classification. Fuzzy clustering is a type of nonexclusive classification in which a pattern is assigned a degree of belongingness to each cluster in a partition and is explained in Section 3.3.8.

b. Intrinsic versus extrinsic. An intrinsic classification uses only the proximity matrix to perform the classification. Intrinsic classification is called “unsupervised learning” in pattern recognition because no category labels denoting an a priori partition of the objects are used. (See Appendix A for an introduction to pattern recognition.) Extrinsic classification uses category labels on the objects as well

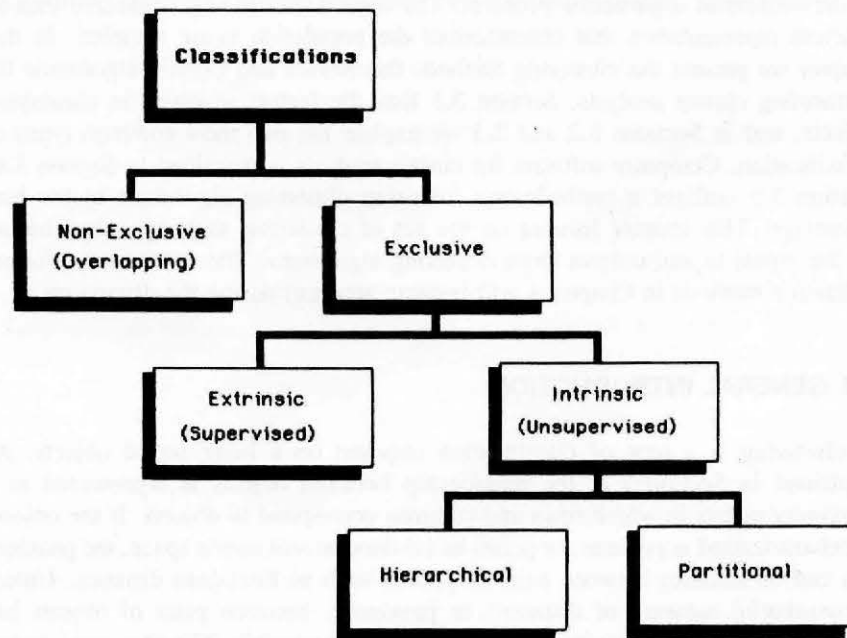


Figure 3.1 Tree of classification types.

as the proximity matrix. The problem is then to establish a discriminant surface that separates the objects according to category. In other words, an extrinsic classifier relies on a "teacher," whereas an intrinsic classifier has only the proximity matrix.

One way to evaluate an intrinsic classification is to see how the cluster labels, assigned to objects during clustering, match the category labels, assigned a priori. For example, suppose that various indices of personal health were collected from smokers and nonsmokers. An intrinsic classification would group the individuals based on similarities among the health indices and then try to determine whether smoking was a factor in the propensity of individuals toward various diseases. An extrinsic classification would study ways of discriminating smokers from nonsmokers based on health indices. We are concerned only with intrinsic classification in this book; intrinsic classification is the essence of cluster analysis.

c. *Hierarchical versus partitional.* Exclusive, intrinsic classifications are subdivided into hierarchical and partitional classifications by the type of structure imposed on the data. A hierarchical classification is a nested sequence of partitions and is explained in Section 3.2, whereas a partitional classification is a single partition and is defined in Section 3.3. Thus a hierarchical classification is a special sequence of partitional classifications. We will use the term *clustering* for an exclusive, intrinsic, partitional classification and the term *hierarchical clustering* for an exclusive, intrinsic, hierarchical classification. Sneath and Sokal (1973) apply the acronym SAHN (Sequential, Agglomerative, Hierarchical, Nonoverlapping) to exclusive, intrinsic, hierarchical, agglomerative algorithms. The differences and similarities between algorithms for generating these two types of classifications are the topics of this chapter.

Several algorithms can be proposed to express the same exclusive, intrinsic classification. One frequently uses an algorithm to express a clustering method, then examines various computer implementations of the method. The primary algorithmic options in common use are explained below.

1. *Agglomerative versus divisive.* An agglomerative, hierarchical classification places each object in its own cluster and gradually merges these atomic clusters into larger and larger clusters until all objects are in a single cluster. Divisive, hierarchical classification reverses the process by starting with all objects in one cluster and subdividing into smaller pieces. Thus this option corresponds to a choice of procedure rather than to a different kind of classification. Partitional classification can be characterized in the same way. A single partition can be established by gluing together small clusters (agglomerative) or by fragmenting a single all-inclusive cluster (divisive).
2. *Serial versus simultaneous.* Serial procedures handle the patterns one by one, whereas simultaneous classification works with the entire set of patterns at the same time (see Clifford and Stephenson, 1975).
3. *Monothetic versus polythetic.* This option is most applicable to problems in taxonomy, where the objects to be clustered are represented as patterns, or

points in a space. A monothetic clustering algorithm uses the features one by one, whereas, a polythetic procedure uses all the features at once. For example, a different feature can be used to form each partition in a hierarchical classification under a monothetic algorithm. We will consider only polythetic algorithms.

4. *Graph theory versus matrix algebra.* What is the appropriate mathematical formalism for expressing a clustering algorithm? We will express some algorithms in terms of graph theory, using properties such as connectedness and completeness to define classifications, and express other algorithms in terms of algebraic constructs, such as mean-square-error. The choice is one of clarity, convenience, and personal choice. When implementing an algorithm on a computer, attention must be paid to questions of computational efficiency. This issue is not related to human understanding of the classification method. Some algorithms have convenient expressions under both options.

3.2 HIERARCHICAL CLUSTERING

A hierarchical clustering method is a procedure for transforming a proximity matrix into a sequence of nested partitions. A hierarchical clustering algorithm is the specification of steps for performing a hierarchical clustering. It is often convenient to characterize a hierarchical clustering method by writing down an algorithm, but the algorithm should be separated from the method itself. In addition to defining algorithms and methods in this section, we define the type of mathematical structure a hierarchical clustering imposes on data and describe ways of viewing that structure.

First comes the notion of a sequence of nested partitions. The n objects to be clustered are denoted by the set $\mathcal{X}$.

$$\mathcal{X} = \{x_1, x_2, \dots, x_n\}$$

where x_i is the i th object. A partition, $\mathcal{C}$, of $\mathcal{X}$ breaks $\mathcal{X}$ into subsets $\{C_1, C_2, \dots, C_m\}$ satisfying the following:

$$C_i \cap C_j = \Phi \quad \text{for } i \text{ and } j \text{ from } 1 \text{ to } m, \quad i \neq j$$

$$C_1 \cup C_2 \cup \dots \cup C_m = \mathcal{X}$$

In this notation, “ $\cap$ ” stands for set intersection, “ $\cup$ ” stands for set union, and Φ is the empty set. A clustering is a partition; the components of the partition are called clusters. Partition $\mathcal{B}$ is nested into partition $\mathcal{C}$ if every component of $\mathcal{B}$ is a proper subset of a component of $\mathcal{C}$. That is, $\mathcal{C}$ is formed by merging components of $\mathcal{B}$. For example, if the clustering $\mathcal{C}$ with three clusters and the clustering $\mathcal{B}$ with five clusters are defined as follows, then $\mathcal{B}$ is nested into $\mathcal{C}$. Both $\mathcal{C}$ and $\mathcal{B}$ are clusterings of the set of objects $\{x_1, x_2, \dots, x_{10}\}$.

$$\mathcal{C} = \{(x_1, x_3, x_5, x_7), (x_2, x_4, x_6, x_8), (x_9, x_{10})\}$$

$$\mathcal{B} = \{(x_1, x_3), (x_5, x_7), (x_2), (x_4, x_6, x_8), (x_9, x_{10})\}$$

Neither $\mathcal{C}$ nor $\mathcal{B}$ is nested into the following partition, and this partition is not nested into $\mathcal{C}$ or $\mathcal{B}$.

$$\{(x_1, x_2, x_3, x_4), (x_5, x_6, x_7, x_8), (x_9, x_{10})\}$$

A hierarchical clustering is a sequence of partitions in which each partition is nested into the next partition in the sequence. An *agglomerative* algorithm for hierarchical clustering starts with the disjoint clustering, which places each of the n objects in an individual cluster. The clustering algorithm being employed dictates how the proximity matrix should be interpreted to merge two or more of these trivial clusters, thus nesting the trivial clustering into a second partition. The process is repeated to form a sequence of nested clusterings in which the number of clusters decreases as the sequence progresses until a single cluster containing all n objects, called the conjoint clustering, remains. A *divisive* algorithm performs the task in the reverse order.

A picture of a hierarchical clustering is much easier for a human being to comprehend than is a list of abstract symbols. A *dendrogram* is a special type of tree structure that provides a convenient picture of a hierarchical clustering. A dendrogram consists of layers of nodes, each representing a cluster. Lines connect nodes representing clusters which are nested into one another. Cutting a dendrogram horizontally creates a clustering. Figure 3.2 provides a simple example. Section 3.2.2 explains the role of dendrograms in hierarchical clustering.

Other pictures can also be drawn to visualize a hierarchical clustering (Kleiner and Hartigan, 1981; Friedman and Rafsky, 1981; Everitt and Nicholls, 1975). Information other than the sequence in which clusterings appear will be of interest. The level, or proximity value, at which a clustering is formed can also be recorded. If objects are represented as patterns, or points in a space, the centroids of the clusters can be important, as well as the spreads of the clusters.

Two specific hierarchical clustering methods are now defined called the *single-link* and the *complete-link* methods. Section 3.2.1 explains algorithms for these two commonly used hierarchical clustering methods. The sequences of clusterings created by these two methods depend on the proximities only through their rank

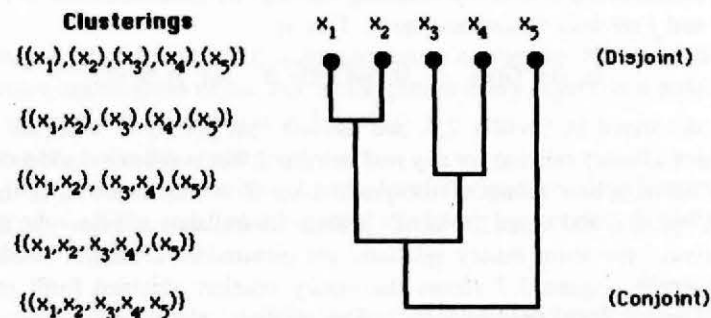


Figure 3.2 Example of dendrogram.

order. Thus we first assume an ordinal scale for the proximities and use graph theory to express algorithms. Single-link and complete-link hierarchical methods are not limited to ordinal data. Sections 3.2.4 and 3.2.5 examine algorithms for these two methods in terms of interval and ratio data. The effects of proximity ties on hierarchical clustering are discussed in Section 3.2.6, while algorithms defined for single-link and complete-link clustering are generalized in Sections 3.2.7 and 3.2.9 to establish new clustering methods. The issues in determining whether or not a hierarchical classification is appropriate for a given proximity matrix are postponed until Chapter 4.

3.2.1 Single-Link and Complete-Link Algorithms from Graph Theory

We begin with a symmetric $n \times n$ proximity matrix $\mathcal{D} = [d(i, j)]$, as defined in Section 2.2. The $n(n - 1)/2$ entries on one side of the main diagonal are assumed to contain a permutation of the integers from 1 to $n(n - 1)/2$ with no ties. That is, the proximities are on an ordinal scale. We take the proximities to be dissimilarities; $d(1, 2) > d(1, 3)$ means that objects 1 and 3 are more like one another than are objects 1 and 2.

Example 3.1

An example of an ordinal proximity matrix for $n = 5$ is given as matrix $\mathcal{D}_1$.

$$\mathcal{D}_1 = \begin{matrix} & \begin{matrix} x_1 & x_2 & x_3 & x_4 & x_5 \end{matrix} \\ \begin{matrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{matrix} & \begin{bmatrix} 0 & 6 & 8 & 2 & 7 \\ 6 & 0 & 1 & 5 & 3 \\ 8 & 1 & 0 & 10 & 9 \\ 2 & 5 & 10 & 0 & 4 \\ 7 & 3 & 9 & 4 & 0 \end{bmatrix} \end{matrix}$$

A *threshold graph* is an undirected, unweighted graph on n nodes without self-loops or multiple edges. Each node represents an object. See Appendix G for a brief review of terms in graph theory. A threshold graph $G(v)$ is defined for each dissimilarity level v by inserting an edge (i, j) between nodes i and j if objects i and j are less dissimilar than v . That is,

$$(i, j) \in G(v) \quad \text{if and only if} \quad d(i, j) \leq v$$

As discussed in Section 2.2, we assume that $d(i, i) = 0$ for all i . Thus $G(v)$ defines a binary relation for any real number v that is reflexive and symmetric. A binary relation is a subset of the product set $\mathcal{X} \times \mathcal{X}$, where $\mathcal{X}$ is the set of objects. Objects x_i and x_j are “related” if their dissimilarity is below the threshold v . Reflexive, symmetric binary relations are pictured in a natural fashion by a threshold graph. Figure 3.3 shows the binary relation obtained from proximity matrix $\mathcal{D}_1$ above for a threshold of 5. The symbol “*” in position (i, j) of the matrix means that the pair (x_i, x_j) belongs to the binary relation.

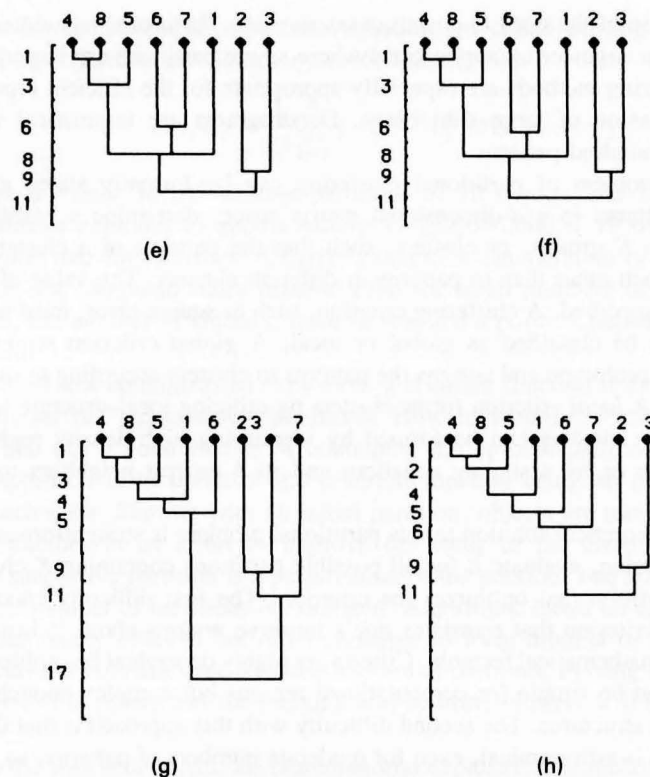


Figure 3.15 (continued)

3.3 PARTITIONAL CLUSTERING

Hierarchical clustering techniques organize the data into a nested sequence of groups. An important characteristic of hierarchical clustering methods is the visual impact of the dendrogram, which enables a data analyst to see how objects are being merged into clusters or split at successive levels of proximity. The data analyst can then try to decide whether the entire dendrogram describes the data or can select a clustering, at some fixed level of proximity, which makes sense for the application in hand. We refer to nonhierarchical clustering methods as *partitional* clustering methods. They generate a single partition of the data in an attempt to recover natural groups present in the data. Both clustering strategies have their appropriate domains of applications. Hierarchical clustering methods generally require only the proximity matrix among the objects, whereas partitional techniques expect the data in the form of a pattern matrix. It is generally assumed that the features have been measured on a ratio scale.

Hierarchical techniques are popular in biological, social, and behavioral sci-

ences because of the need to construct taxonomies. Partitional techniques are used frequently in engineering applications where single partitions are important. Partitional clustering methods are especially appropriate for the efficient representation and compression of large data bases. Dendrograms are impractical with more than a few hundred patterns.

The problem of partitional clustering can be formally stated as follows. Given n patterns in a d -dimensional metric space, determine a partition of the patterns into K groups, or clusters, such that the patterns in a cluster are more similar to each other than to patterns in different clusters. The value of K may or may not be specified. A clustering criterion, such as square-error, must be adopted. Criteria can be classified as global or local. A *global* criterion represents each cluster by a prototype and assigns the patterns to clusters according to most similar prototypes. A *local* criterion forms clusters by utilizing local structure in the data. For example, clusters can be formed by identifying high-density regions in the pattern space or by assigning a pattern and its k nearest neighbors to the same cluster.

The theoretical solution to this partitional problem is straightforward. Simply select a criterion, evaluate it for all possible partitions containing K clusters, and pick the partition that optimizes the criterion. The first difficulty encountered is selecting a criterion that translates one's intuitive notions about "cluster" into a reasonable mathematical formula. Criteria are highly dependent on problem parameters and must be simple for computational reasons but complex enough to reflect various data structures. The second difficulty with this approach is that the number of partitions is astronomical, even for moderate numbers of patterns, so evaluating even the simplest criterion over all partitions is impractical.

Let $S(n, K)$ denote the number of clusterings of n objects into K clusters. The order of the objects in each cluster and the order of the clusters themselves are immaterial. Empty clusters are not counted. A partial difference equation can be written for $S(n, K)$ as follows. Suppose that all clusterings of $n - 1$ objects have been listed. A clustering of n objects can be formed from this list in two ways.

1. The n th object can be added as a singleton cluster to each member of the list with exactly $(K - 1)$ clusters.
2. The n th object can be added to each cluster of any member of the list with exactly K clusters.

Thus

$$S(n, K) = S(n - 1, K - 1) + KS(n - 1, K)$$

The boundary conditions on this equation are

$$S(n, 1) = 1, S(n, n) = 1, S(n, K) = 0 \quad \text{if } K > n$$

The solution to this equation for $S(n, K)$ requires that values $\{S(j, p)\}$ be known for the set $\{(j, p) : 1 \leq j \leq n - 2, 1 \leq p \leq K\}$.

Solutions to the partial difference equation are called Stirling numbers of the second kind (Fortier and Solomon, 1966; Jensen, 1969):

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

There are only 34,105 distinct partitions of 10 objects into four clusters, but this number explodes to approximately 11,259,666,000 if 19 objects are to be partitioned into four clusters. Clearly, exhaustive enumeration of all possible partitions is not computationally feasible even for small numbers of patterns. In addition, K , the number of clusters, must be selected a priori. Chapter 4 discusses this problem.

To avoid this combinatorial explosion, a criterion function is evaluated only for a small set of "reasonable" partitions. How to identify a small subset of partitions that has a good chance of containing the optimal partition? The most common approach is to optimize the criterion function using an iterative, hill-climbing technique. Starting with an initial partition, objects are moved from one cluster to another in an effort to improve the value of the criterion function. Thus each successive partition is a perturbation of the previous one and, therefore, only a small number of partitions is examined. Algorithms based on this technique are computationally efficient but often converge to local minima of the criterion function. Several heuristics for choosing the initial partition, moving objects from one cluster to the other, and for merging and splitting clusters will be discussed later.

Another way of avoiding the combinatorial explosion is somehow to identify and reject a large number of partitions which are not likely to be of interest. Jensen (1969) uses a dynamic programming approach to eliminate many partitions and is still able to achieve an optimal solution. A significant computational savings is realized, especially for large clustering problems, at the expense of algorithmic complexity. For example, to partition 19 objects into four clusters, less than 2% of the total number of partitions need to be evaluated using the dynamic programming formulation. Even this reduction is not enough to make this approach computationally feasible for practical clustering problems. Several related approaches are described in the literature (Edwards and Cavalli-Sforza, 1965; Vinod, 1969; Rao, 1971; Koontz et al., 1975; Lefkovitch, 1980).

There is no single "best" criterion for obtaining a partition because no precise and workable definition of "cluster" exists. Clusters can be of arbitrary shapes and sizes in a multidimensional pattern space. Each clustering criterion imposes a certain structure on the data, and if the data happen to conform to the requirements of a particular criterion, the true clusters are recovered. Only a small number of independent clustering criteria can be understood both mathematically and intuitively. Thus the hundreds of criterion functions proposed in the literature are related and the same criterion appears in several disguises. Shaffer et al. (1979) demonstrate the similarity of a mode-seeking partitional algorithm (Kittler, 1976) and the MST-based algorithm of Zahn (1971). Similarly, Urquhart (1982) shows that partitions obtained from a relative neighborhood graph are identical to those

generated by mutual near-neighbor clustering (Gowda and Krishna, 1978). The literature of cluster analysis is spread so widely and over so many areas of science that a single criterion function is rediscovered repeatedly.

In this section we present some of the most common partitional clustering methods. Several popular criteria are versions of square-error and are discussed in Sections 3.3.1 to 3.3.3. Another global criterion obtains a partition by fitting a mixture density model to the patterns (Section 3.3.4). These clustering criteria create clusters having hyperellipsoidal shapes. Sections 3.3.5 to 3.3.7 describe several partitional clustering methods based on local criteria of density or mode estimation, graph connectivity, and near-neighbor relationships. Finally, Section 3.3.8 briefly covers the topic of fuzzy clustering, where each object is permitted to belong to more than one cluster with a grade of membership.

Most of the partitional clustering techniques presented here implicitly assume continuous-valued feature vectors so that the patterns can be viewed as being embedded in a metric space. If the features are on a nominal or ordinal scale, Euclidean distances and cluster centers are not very meaningful, so hierarchical clustering methods are normally applied. Wong and Wang (1979) proposed a clustering algorithm for discrete-valued data. The approach is similar to the mode estimation procedure for continuous data but approximates the high-order discrete probability distribution by a second-order product and uses Hamming distance (Section 2.2) between patterns.

The technique of conceptual clustering or learning from examples (Michalski and Stepp, 1983) can be used with objects represented by nonnumeric or symbolic descriptors. The objective here is to group objects into conceptually simple classes. Clustering of trains using attributes such as number of cars, number of wheels, colors of wheels, and number of items carried, and the clustering of microcomputers using attributes such as CPU speed, memory size, and type of processor are more appropriately handled by associating each cluster with a simple "concept." Concepts are defined in terms of attributes. For example, in the train classification problem, trains with two red cars is a concept. Objects are arranged into a hierarchy of classes described by concepts.

3.3.1 Square-Error Clustering Criteria

The most commonly used partitional clustering strategy is based on the square-error criterion. The general objective is to obtain that partition which, for a fixed number of clusters, minimizes the square-error. Ward's method of hierarchical clustering (Section 3.2.7) uses square-error in a different way. Minimizing square-error, or within-cluster variation, will be shown to be equivalent to maximizing the between-cluster variation.

Suppose that the given set of n patterns in d dimensions has somehow been partitioned into K clusters $\{C_1, C_2, \dots, C_K\}$ such that cluster C_k has n_k patterns and each pattern is in exactly one cluster, so that

$$\Phi_f = 1 - \frac{2}{K-1} \sum_{k=1}^{K-1} \sum_{j=k+1}^K I(f_{C_k} \cap f_{C_j})$$

where K is the number of clusters and $I(f_{C_k} \cap f_{C_j})$ is the intersection of two fuzzy sets (clusters) defined as follows:

$$I(f_{C_k} \cap f_{C_j}) = \frac{1}{n} \sum_{\mathbf{x} \in \mathcal{X}} \min [f_{C_k}(\mathbf{x}), f_{C_j}(\mathbf{x})]$$

The minimum value of Φ_f is 0, which represents maximum fuzziness, and the maximum value of Φ_f is 1, which corresponds to a nonfuzzy partition. The problem of fuzzy clustering is to find that partition which maximizes Φ_f . The basic steps in a fuzzy partitional clustering algorithm are given below.

FUZZY PARTITIONAL CLUSTERING ALGORITHM

- Step 1.** Select an initial partition $\{C_k\}_{k=1}^K$.
Repeat steps 2 to 4 until the cluster memberships stabilize.
- Step 2.** Compute the membership functions $\{f_{C_k}(\mathbf{x}_j)\}$.
- Step 3.** Compute the criterion function Φ_f .
- Step 4.** Reclassify patterns to improve Φ_f .

The output of a fuzzy algorithm not only includes a partition but also additional information in the form of membership values. However, the new information provided by the membership values must be interpreted by the data analyst. In summary, fuzzy clustering is an interesting concept that includes most partitional clustering algorithms as special cases. However, its superiority to ordinary clustering has yet to be demonstrated in applications.

3.4 CLUSTERING SOFTWARE

Someone interested in applying clustering techniques has plenty of software to choose from. As we reported earlier, there is no shortage of clustering algorithms, and most of them have been implemented to run on a variety of computers. One of the main reasons for the large number of "different" clustering programs available is that most researchers and research groups put all their trust in the algorithm they have developed even though it is very similar to existing algorithms. In addition, documented and tested clustering packages were not available until recently, so users had to write their own software. The paper by Johnston et al. (1979) illustrates how the "law of serendipity" can lead to a "new" clustering algorithm. During the coding of the mode-seeking algorithm by Koontz et al. (1976), Johnston et al., by mistake, modified one of the expression for the relative density of a pair of patterns. This led to a new clustering algorithm which provided better clustering for "uniform, touching" clusters than the original!

A clustering technique imposes a particular structure on the given data. Since the purpose of cluster analysis is to reveal structure or "pattern" in large quantities of numerical data, one should not put too much faith in interpreting a single clustering. An agreement in the clusterings generated by two or more different clustering methods increases one's confidence in accepting the imposed structure as the true structure present in the data. We again emphasize the difference between a clustering method and a clustering algorithm. An algorithm is a particular way of implementing a method. Thus the square-error clustering method can be implemented in several ways, as illustrated by programs such as ISODATA, CLUSTER, FORGY, and K-MEANS. These algorithms might not generate the same clusterings when applied to a data set. So it is important that a user of cluster analysis should be aware of the true differences between the various clustering programs.

Blashfield et al. (1982) provide excellent coverage of the availability of cluster analysis software. They even suggest that "there may be as many different programs in existence to perform cluster analysis as there are users." They list three major categories of cluster analysis software which are summarized briefly below.

1. *Collection of subroutines and algorithms.* These are usually programs developed by a research group which can be obtained without any charge. Some of these collections have appeared in the books by Anderberg (1973) and Hartigan (1975). There is generally no guarantee on the quality of such software. A few subroutines are available through a commercial organization called IMSL (International Mathematical and Statistical Library of scientific subroutines) which are well maintained. Users have to write their own driver programs to execute IMSL routines.
2. *General statistical packages that contain clustering methods.* Several of the well-known statistical packages, such as the BMDP series, SAS, and OSIRIS, have a limited collection of clustering programs. These packages are widely available at university computer centers, are well maintained, and come with good documentation. New versions of these packages now run on personal computers also.
3. *Cluster analysis packages.* The best known and most versatile clustering package available is a collection of programs written in Fortran called CLUSTAN. CLUSTAN contains a comprehensive choice of clustering techniques. This package can be obtained from: Computer Center, University College, London, 19 Gordon Street, London, WC1H 0AH. Other clustering packages are NT-SYS, CLUS, TAXON, BC-TRY, S, and CLAB. A package called ICICLE performs hierarchical clustering and displays the results as Icicle plots. It can be obtained from: Computer Information Services Group, Room 2F128, AT&T Bell Laboratories, Murray Hill, NJ 07974. These packages offer more flexibility than a general statistical package. Some of the options available to the user include data screening, data transformation, a wide choice of similarity measures, and cluster diagnostics.

In summary, there is no “complete” cluster analysis package available in the market. The software described above implements only the most popular techniques. Perhaps this is appropriate in light of the large number of clustering methods reported in the literature. The packages available tend to favor hierarchical techniques over the partitional techniques because of the predominance of clustering users in social, behavioral, and biological sciences. Since there is no “best” clustering program, serious users of clustering must be prepared to implement algorithms which are more suited to their data than those available commercially. Software for pattern recognition algorithms, such as ISPAHAN (Gelsema, 1980), and multidimensional scaling, such as MDSCAL, will also be useful to a user of cluster analysis.

3.5 CLUSTERING METHODOLOGY

This chapter has explained the details of clustering algorithms. Chapter 2 presented ways of inspecting and representing data. Chapter 4 reviews a number of techniques for validating the results of clustering algorithms. How does it all go together? This section proposes a generic methodology for focusing the tools developed in this book into a procedure for analyzing data. The needs and special circumstances encountered in individual problems may make parts of the methodology unnecessary and other parts impossible. However, it is worthwhile to take a step back from the details and look at the big picture as is done in Section 3.5.1. Comparative analyses of clustering methods that summarize experimental facts about cluster analysis and aid in choosing strategies and tactics are reviewed in Section 3.5.2.

This methodology is more of a proposal than an accepted standard. Case studies that follow this methodology are difficult to find because each application has its special needs. We recommend this methodology for the serious practitioner.

3.5.1 Exploratory Data Analysis

Figure 3.35 relates the major steps to be considered when undertaking an exploratory data analysis whose central component is a cluster analysis. We see

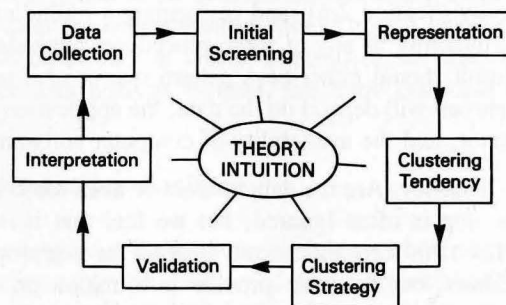


Figure 3.35 Clustering methodology.