Pattern Recognition Pergamon Press 1976, Vol. 8, pp. 247–260. Printed in Great Britain

CLUSTERING TECHNIQUES: THE USER'S DILEMMA

RICHARD DUBES and ANIL K. JAIN
Department of Computer Science, Michigan State University, East Lansing, MI 48824, U.S.A.

(*Received* 24 *October* 1975)

Abstract—Numerous papers on clustering techniques and their applications in engineering, medical, and biological areas have appeared in pattern recognition literature during the past decade. This paper attempts to set some guidelines for a potential user of a clustering technique. We examine eight clustering programs which are representative of the various available techniques and compare their performances from several points of view. A formal comparative analysis is also performed with a portion of Munson's handprinted character data set. We believe that an understanding of the intrinsic characteristics of a clustering technique is essential to the intelligent application of the technique. Further, the output of a clustering program, along with whatever information a user may have about the data set, should be used together to form hypotheses about the structure of the data set.

Clustering technique Patterns Features Squared error Distance measures Dendrogram Similarity matrix Hierarchical clustering Minimum spanning tree Admissibility criteria

1. INTRODUCTION

Clustering techniques attempt to group points in a multidimensional space in such a way that all points in a single group have a natural relation to one another and points not in the same group are somehow different. The points themselves, in pattern recognition terminology, are called "patterns" or OTU's (operational taxonomic units), while the axes of the pattern space are called "features" or "measurements." The user of a clustering technique is trying to understand a set of data and to uncover whatever structure resides in the data. Clustering techniques are tools for discovery rather than ends in themselves and should permit the user to form statistical questions for future studies. Their application and their interpretation are subjective, depending on the experience and perspicacity of the user. Anderberg[1] and Everitt[2] provide excellent treatments of clustering methodology.

The subjective nature of the clustering problem precludes a realistic mathematical comparison of all clustering techniques. Anderberg[1] explains the difficulties inherent in attempting to fit the intuitive nature of clustering techniques into any useful mathematical framework. Nevertheless, a potential user needs some rational basis for choosing among clustering programs. This paper examines eight clustering programs and compares their performances from several points of view. The objective is not to choose a "best" clustering technique or program. Such a task would be fruitless and contrary to the very nature of clustering. Rather, the peculiarities of the techniques and programs are presented, emphasizing those aspects most important to the user. A formal comparative analysis is also performed with a particular data base.

This paper carefully distinguishes between clustering techniques and clustering programs. A clustering technique is a general strategy for forming clusters, such as minimizing squared error or cutting a minimum spanning tree. A clustering program is the actual computer program that implements a strategy, including all the heuristic tactics and *ad hoc* procedures inherent in a clustering program. For example, four programs that try to minimize squared error are investigated. They differ only in the tactics employed. This paper compares both the computational aspects of the programs and the theoretical bases for the techniques.

2. DEFINITIONS OF CLUSTERING TECHNIQUES AND PROGRAMS

The eight clustering programs treated in this paper represent three clustering techniques and provide a representative sample of the many clustering programs suggested in the literature. The general nature of each program is described in this section.

Four of the eight clustering programs exemplify the first clustering technique—minimizing squared error. These squared-error programs differ both in computational details and in the approach taken to minimize the squared error. Two of the eight programs illustrate the second technique—hierarchical clustering. The last two programs are very different examples of the third technique—graph-theoretic clustering. The outputs of the eight programs are not the same, so a direct comparison isn't possible. However, a user is faced with the same dilemma—choosing a program even though the outputs of the alternatives aren't comparable. The results of this paper should help resolve this dilemma.

2.1 *User options*

All the programs considered here have a scaling option. The user can choose to normalize the patterns to be clustered by making the sample means of all features zero and the sample variances one. This involves moving the origin of the pattern space to the centroid of all patterns and stretching or compressing each axis in the pattern space. Moving the origin has no significant effect on the clustering, but normalizing the spreads of the individual features assigns equal weight to all features and tends to transform the cloud of patterns into an hypersphere.

A second *ad hoc* choice the user must make is the selection of a distance metric, or measure of dissimilarity between two patterns. Since Euclidean distance is familiar and is invariant under translations of the origin and rotations of the pattern space, it was adopted for this study. It is a simple matter to insert any Minkowski metric[1] into any of the programs. The inadequacies of Euclidean distance measures in clustering have been discussed by Hall *et al.*[3]

2.2 *Computational cost*

The user of any computer program is interested in the cost of the program before it is actually used. The two most costly aspects of clustering programs are run time (CPU sec) and central memory usage (CM word–hr). The squared-error clustering programs are iterative with internal stopping criteria. The other programs involve several data-dependent computational features. Thus, it is impossible to accurately predict the cost of running any of the programs.

Convergence and termination become problems with the squared-error programs and have been treated by Anderberg,[1] who shows that the methods on which the squared-error programs are based actually converge. The programs prevent cycling through the same sequence of clusterings repeatedly. The user also supplies a limit on the number of iterations allowed. In practice, the programs usually terminate naturally, which means that two successive iterations find the same clustering.

The four squared-error programs require that a pattern matrix representation of the data exists. By contrast, the other four programs process the pattern matrix into a dissimilarity matrix and use it alone for clustering. Three of the four squared-error programs hold the entire matrix in core storage throughout the run and one—program WISH—stores only one pattern at a time.

2.3 *Type of output*

A clustering program is a means for understanding a set of patterns through creation of a mental picture. But what sort of picture does the user actually obtain? Most desirable is a two-dimensional representation such as that proposed by Sammon,[4] which is essentially Kruskal's multidimensional scaling technique,[5,6] an example of a method for studying intrinsic dimensionality.[7] Such techniques require nonlinear transformations of the patterns and the results can be difficult to interpret in terms of the original features. Unless one knows something about the data structure being sought, the validity of any two-dimensional representation obtained from a nonlinear transformation is very difficult to assess. No prior knowledge about data structure is assumed in this paper.

A method for deriving a linear transformation from the pattern space to a two-dimensional space is provided by the Karhunen–Loeve expansion.[8] When category information exists, discriminant analysis[9,10] suggests another linear transformation. However, the power of a linear transformation is limited and there is simply no way to accurately represent data in two dimensions unless the data have peculiar properties.

Abandoning the two-dimensional representation leaves the user with the difficult task of interpreting tables of numbers and creating an image from them.[1] The primary advantage of an hierarchical clustering program is the picture of the data it generates. Humans can assimilate information from pictures better than from tables of numbers.

The basic output of a squared-error clustering program is a listing of the cluster centers, a table of the distances between them, the squared-error for each cluster, and the membership of each cluster. When a category label (pattern class label) can be assigned to each pattern *a priori*, as in the empirical study in Section 3.2, a table showing the number of patterns from each category in each cluster is also computed. Such a table is extremely helpful in judging the utility of the clustering because the user often searches for a clustering which "explains" the categories. A clustering in which each cluster is dominated by a single category is also justification for the features being employed.

Several other statistics that have a statistical meaning with an underlying Gaussian model are also usually computed. For example, an analysis of variance can be preformed for each feature to determine which features best distinguish among the clusters. A discriminant analysis can also be performed to determine if the clusters are "significantly" different. Only the basic output is reported here.

2.4 *Squared-Error clustering programs*

The ubiquity of statistical procedures based on underlying Gaussian models makes clustering programs that minimize a squared error criterion very popular. Such programs try to define clusters that are hyperellipsoidal in shape and are sometimes characterized as nonparametric methods for fitting mixtures of Gaussian distributions to the data, also called"unsupervised learning."[11]

Let the i^{th} pattern, $i = 1, \ldots, n$ from the data set under study be written as

$$x_i = (x_{i1}\ x_{i2}\ \text{---}\ x_{iN})^T.$$

The number of patterns, n, is assumed significantly larger than the number of features, N. A clustering is a partition $[C_1, C_2, \ldots, C_K]$ of the integers $[1, 2, \ldots, n]$ that assigns each pattern a single cluster label. The patterns corresponding to the integers in C_k form the k^{th} cluster, whose center is:

$$c_k = (c_{k1}\, c_{k2} \ldots c_{kN})^T,$$

where

$$c_{kj} = (1/M_k) \sum_{i \in C_k} x_{ij}$$

and M_k is the cardinality of C_k, or the number of patterns in cluster k. Thus, a cluster center is simply the centroid, or sample mean, of all patterns in the cluster. The squared error for cluster k is:

$$e_k^2 = \sum_{i \in C_k} (x_i - c_k)^T (x_i - c_k),$$

and the squared error for the clustering is:

$$E_K^2 = \sum_{k=1}^{K} e_k^2.$$

This squared error can be expressed in many ways, such as the sum of the "within" and "between" squared errors used in discriminant analysis.[9,10] The objectives of a squared-error clustering program are to define, for a given K, a clustering that minimizes E_K^2 and to find a suitable K, much smaller than n. An exhaustive search being computationally unfeasible,[1,p.3] the various squared-error programs implement different tactics for searching through the possible clusterings. All programs try to find a local minimum of E_K^2 and the user hopes that the local minimum coincides with the global minimum.

The programs investigated in this paper can be compared on four factors:

(1) manner in which the next clustering is chosen, or cluster updating;

(2) criterion for creating new clusters, or increasing K;

(3) criterion for deleting and merging clusters, or decreasing K and identifying outliers;

(4) initialization procedure, or establishing the first clustering.

Only factor (1) is characteristic of the program, the remaining factors being heuristic and somewhat interchangeable among programs. However, these heuristic procedures often spell success or failure for a clustering program, irrespective of the tactic chosen in (1) and are crucial elements in the clustering program. Realize that the squared error tends to decrease as the number of clusters increases so one can mathematically minimize E_K^2 only for fixed K. The heuristics in factors (2) and (3) allow the program to select a "natural" or "suitable" number of clusters, subject to user-imposed restrictions.

The four squared-error programs are now defined and compared.

*Since the programming technique can have a profound effect on the output of a clustering program, the names used here suggest only the intent of the method.

2.4.1 *Forgy's method.** The simplest of the squared-error programs is FORGY, which implements Forgy's method.[1] A simplified flowchart is given in Fig. 1. The heart of Forgy's method is the inner loop in Fig. 1 which establishes the way in which clusters are updated. Given a set of cluster centers, the cluster label of the closest cluster center is assigned to each pattern. The cluster centers are then recomputed as sample means, or centroids, of all patterns having the same cluster label.

A new cluster is created in the inner loop when a pattern is found that is sufficiently removed from the existing structure. Let $d_k(i)$ be the distance between pattern i and cluster center k. Let $\bar{d}(i)$ be the average distance from pattern i to all K cluster centers.

$$\bar{d}(i) = (1/K) \sum_{k=1}^{K} d_k(i).$$

A new cluster is created centered at pattern i if:

$$|d_{k_o}(i) - \bar{d}(i)| \leq |\bar{d}(i)|\, T_F,$$

where k_o is the cluster center closest to pattern i and T_F is a user-supplied threshold between zero and one. The larger T_F, the more new clusters will be created. The inner loop is repeated until either two successive passes through all patterns produce the same cluster-

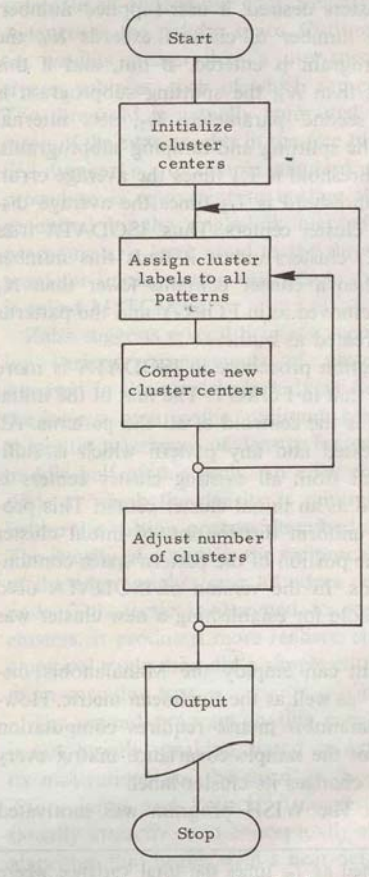


Fig. 1. Flowchart for FORGY and ISODATA.

ing or a user-supplied limit, L_F, on the number of loops has been exceeded. The number of patterns, M_k, in cluster k is then computed for each k and compared to the user-supplied number N_F. If $M_k < N_F$, all patterns in cluster k are removed and henceforth ignored. Thus, such patterns are considered to be outliers. This is the only means available in FORGY for reducing the number of clusters.

FORGY was constructed to be as direct as possible. The initialization procedure follows this philosophy and fixes the intital cluster centers by selecting K_F patterns at random, where K_F is supplied by the user.

2.4.2 *ISODATA.* The ISODATA method[1] is the most famous of the squared-error clustering methods. Clusters are updated as in FORGY (Fig. 1). ISODATA is unique in the heuristics employed to create new clusters while trying to achieve the number of clusters requested by the user. The capability of dividing and merging existing clusters permits ISODATA to recover from poor initial clusterings and to search over a wide range of clusterings.

The positive user-supplied parameters, T_{I1} and T_{I2}, control the splitting apart and lumping together of clusters. The first controls entrance to the splitting and lumping subprograms as follows. Define K_I' as the integer part of $(1 + T_{I1})K_I$ where K_I is the number of clusters desired, a user-supplied number. If the current number of clusters exceeds K_I', the lumping subprogram is entered. If not, and if this number is less than K_I, the splitting subprogram is entered. The second parameter, T_{I2}, sets internal thresholds in the splitting and lumping subprograms. The splitting threshold is T_{I2} times the average error. The lumping threshold is T_{I2} times the average distance between cluster centers. Thus, ISODATA tries to establish K_I clusters unless it finds this number unrealistic. When a cluster contains fewer than N_I patterns, it is removed as in FORGY and the patterns involved are treated as outliers.

The initialization procedure in ISODATA is more elaborate than that in FORGY. The first of the initial cluster centers is the centroid of all the patterns. All patterns are tested and any pattern which is sufficiently removed from all existing cluster centers is itself designated as an initial cluster center. This provides a fairly uniform distribution of initial cluster centers over the portion of the pattern space containing the patterns. In the version of ISODATA used here, the threshold for establishing a new cluster was set internally.

This program can employ the Mahalanobis distance metric[12] as well as the Euclidean metric. However, the Mahalanobis metric requires computation of the inverse of the sample covariance matrix every time a pattern changes its cluster label.

2.4.3 *WISH.* The WISH program was motivated by Wishart's variant on the k-means method.[1] A simplified flowchart is shown in Fig. 2. WISH processes one pattern at a time and never stores the entire pattern matrix in core.

Cluster centers are updated in the "DISPOSE" box in the inner loop immediately after a cluster label is assigned to each pattern. This approach was suggested[13] as a means for estimating multidimensional densities and has been used in other clustering techniques.[14] To see how this works, let $\theta_W > 1$ and $t_W > 1$ be user-supplied* thresholds. The DISPOSE box treats pattern i as follows. If

$$dk_o(i) \leq t_W,$$

where k_o is the number of the cluster center closest to pattern i and $d_{ko}(i)$ is the distance between them, and if pattern i is not already in cluster k_o, pattern i is added to cluster k_o by updating the cluster centers involved. If

$$t_W < dk_o(i) < \theta_W t_W,$$

pattern i is placed on the junkpile, which sets it aside by temporarily removing it from the clustering. The junkpile is forced into the clustering after every iteration through all the patterns. If

$$dk_o(i) \geq \theta_W t_W,$$

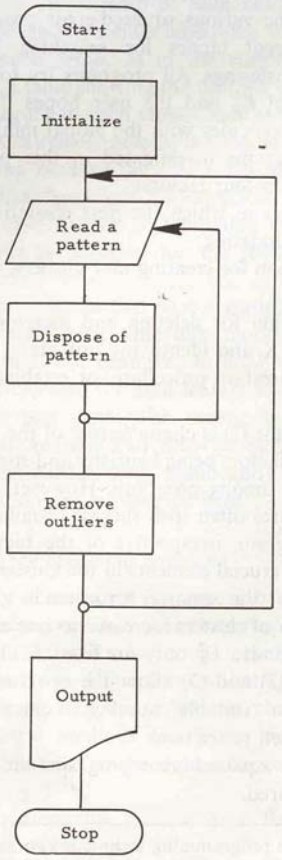


Fig. 2. Flowchart for WISH.

*t_W is computed as T_W times the total variance where T_W is supplied by the user.

a new cluster is formed centered at pattern i. The user specifies the maximum allowable number of clusters. Clusters are deleted as in FORGY.

The initialization procedure is more elaborate than that in FORGY. It identifies cluster centers distributed over the cloud of patterns but retains an element of randomness. Given a user-specified parameter K_W, K_W points in the pattern space, called accumulation points, are established. Each pattern is labeled according to the closest accumulation point and the initial cluster centers are the centroids of the patterns having the same label.

Accumulation points are defined from the sample means $(m_1, m_2, \ldots, m_N)$ and the sample standard deviations $(s_1, s_2, \ldots, s_N)$ for the features. The first accumulation point is $(m_1\ m_2\ \text{---}\ m_N)$ and additional points are selected at random from the lattice with vertices $m_j \pm s_j r_W$ where the user supplies r_W.

2.4.4 *CLUSTER*. The CLUSTER program is unique in that it generates an entire sequence of clusterings which are not hierarchically related. This program is based on the "hill-climbing" technique of Friedman and Rubin[12] and has two phases.

Phase 1 creates a sequence of clusterings containing 2, 3, ..., K_C clusters where K_C is specified by the user. The initial cluster centers in the first two-cluster clustering are the centroid of the patterns and the pattern farthest removed from the centroid, not counting outliers. Given a clustering with K clusters, a clustering with $K + 1$ clusters is formed by identifying the pattern most removed from the clustering as a potential cluster center. Each pattern is then tried in every cluster to minimize squared error. Phase 2 merges clusters two at a time to produce a sequence of clusterings containing K_C-1, K_C-2, ---, 2 clusters.

Phases 1 and 2 are alternated until a pass through both decreases the squared error of none of the clusterings. The best clustering ever achieved for each K is retained.

2.5 *Hierarchical clustering programs*

Hierarchical clustering techniques begin with a triangular dissimilarity matrix whose rows and columns correspond to patterns and whose entries measure dissimilarity between patterns; the larger the entry, the more dissimilar the patterns. The entries are the Euclidean distance between the patterns in the pattern space. The two methods employed are the connectedness (single-link) method, implemented in program SINLNK, and the diameter (complete-link) method,[15] implemented in program COMLNK. Many other methods have been suggested.[16,17]

The output of an hierarchical clustering program is a dendrogram, which is a tree showing a sequence of nested clusterings. An example is given in Fig. 3. This graphical output is the outstanding feature of such programs since several clusterings are represented on the same picture. The number of patterns is limited by computational considerations.

The programs used in this study are based on Johnson's algorithms,[14] which can implement any method by insertion of a single subprogram but which is limited to *ca.* 200 patterns because it requires n^2-n memory locations to store two copies of the triangular dissimilarity matrix. Several solutions to the computational problems posed by hierarchical clustering have been suggested.[1,3] The single-link method is a special case because it is equivalent to cutting a minimum spanning tree[18] (see Section 2.6). This fact has been exploited to produce relatively fast algorithms.[19,20] Although the single-link method has more theoretical justification,[16] the complete-link method is easier to interpret in practice.[21]

2.6 *Graph–theoretic clustering programs*

Not all natural groupings of patterns are globular, or hyperellipsoidal in shape. For example, patterns that are spaced along a straight line or on a sheet in the pattern space are well-structured. The squared-error methods force a Gaussian-based model on such structures and cannot succeed. Graph–theoretic methods provide one means for uncovering unconventional data structures.

Zahn[22] gives an overview of graph–theoretic methods. The basic idea is to generate a minimum spanning tree for the complete graph whose nodes are patterns and whose edge weights are Euclidean distances in the pattern space. Cutting all edges having weights greater than a user-specified threshold creates subtrees, each of which represents a cluster. The threshold is actually computed as the sample mean of the edge weights in the tree plus a user-specified number, r_M, of sample standard deviations. This procedure is equivalent to cutting the dendrogram generated by the single-link hierarchical clustering procedure at a level equal to the threshold. The program for implementing this graph–theoretic method is called MSTCLUS.

Zahn suggests several heuristic tactics for uncovering various arrangements of patterns. The one reported in the empirical study in Section 3.2 finds the longest path in the minimum spanning tree and computes a measure of density for each node in the middle half of this path. An edge connected to the node at which the density is minimum is removed before the cutting process described above is begun. The density of a node is the reciprocal of the average of the edge weights over all edges connected to the node. This tactic is designed to separate touching clusters. It produced more realistic clusterings in the empirical study than did a simple cutting of the minimum spanning tree.

The second program in this category, called JP, is not, strictly speaking, based on graph theory, but its motivations are the same as those proposed by Zahn. Jarvis and Patrick[23] suggested a computationally attractive and conceptually simple clustering algorithm that begins with a near-neighbor table. Let $N(i, j)$ denote the (i, j) entry of the table. $N(i,1)$ is the

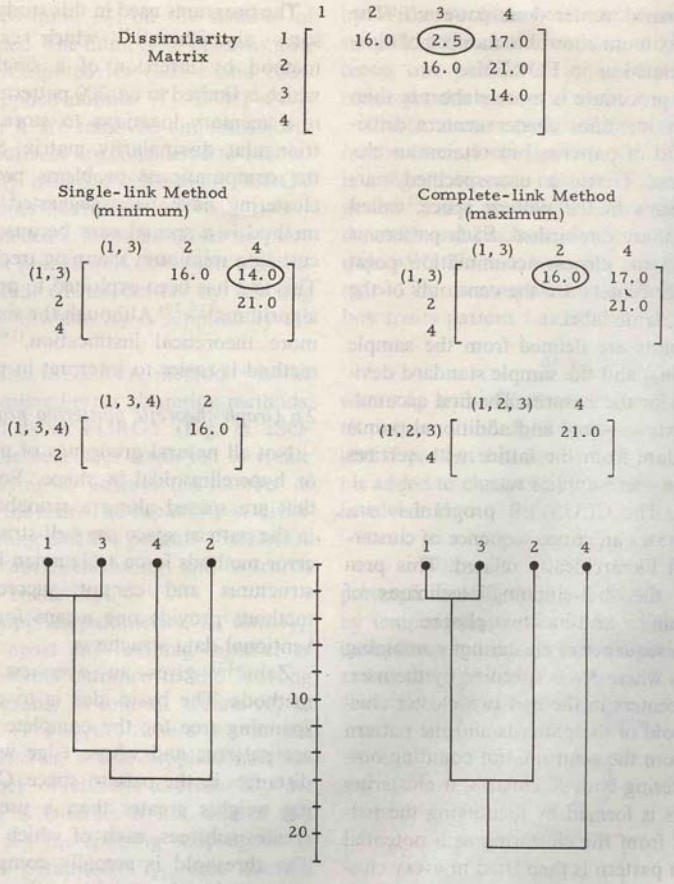


Fig. 3. Examples of dendrograms from single-link and complete-link methods.

cluster label for pattern i. Initially, $N(i,1) = i$ for all i. $N(i,j)$, $j = 2,3,\ldots, M_J$ is the number of the pattern which is the $(j\text{-}1)^{st}$ nearest neighbor of pattern i and M_J is supplied by the user.

The near neighbor table is equivalent to a digraph whose nodes are the patterns and which has M_J-1 edges whose tails are incident to each node and whose heads are at the M_J-1 nearest neighbors of each node. The table is simply a convenient way of storing the digraph. The JP program places patterns i_1 and i_2 in the same cluster if the following conditions are both satisfied.

(1) i_1 is in row i_2 and i_2 is in row i_1;

(2) rows i_1 and i_2 share at least MATCH pattern numbers, excluding the first column of the table.

Incrementing MATCH for a fixed M_J produces an hierarchical sequence of clusterings. Selecting M_J requires some trial and error. For example, setting $M_J = n$ automatically places all patterns in a single cluster.

The intuitive appeal of graph–theoretic techniques is balanced by difficulties in interpreting the results. For instance, a clustering produced by a squared-error program always produces nearly globular clusters, whether or not the natural clusters are globular. What can be said about the shape of clusters produced by graph–theoretic techniques? How is one to know whether a cluster means that the patterns are along a string, or on a sheet, or has any other irregular shape? Clearly, another stage of investigation, such as a study of intrinsic dimensionality,[5,7] is needed for each cluster.

Several heuristic tactics for finding clusters when the data have "necks" or the clusters vary in density have been suggested.[22] Unfortunately, the application of such procedures requires a great deal of prior information. Suppose an heuristic tactic designed for clusters with necks and one designed for touching clusters are both used. How can the two sets of results be compared? Which is better? All studies provided in the literature are two-dimensional so the cluster shapes are obvious.[22,23]

3. METHODS OF COMPARISON

Clustering techniques and clustering programs can be compared from several points of view. Anderberg[1] provides a comprehensive discussion of strategies for performing such comparisons.

One way of comparing clustering techniques is to

define some methematical criterion, such as squared error, select a data base with known properties, such as the iris data,[24] and apply several clustering programs to the data, ranking them according to the value of the criterion.[23] Such an approach is inadequate for several reasons. First, a single criterion cannot summarize all the information that can be gleaned from a clustering. After all, an hierarchical clustering generates an entire sequence of clusterings and squared error clusterings are summarized in several tables. Second, comparing clustering techniques with data having known properties usually means that the data are well-behaved in some sense. Any abilities a clustering technique might have for uncovering pecularities in data are thus masked. Third, such an approach ignores the computational criteria that are of paramount importance to the user, such as difficulty of parameter selection, run time, and storage requirements.

Another method of comparison is to concentrate on the techniques themselves and compare them according to their intrinsic characteristics, such as the types of clusters they are likely to find. The insight into the nature of the clustering technique provided by such an approach is essential to the intelligent application of the technique. However, this approach negates the effect of the heuristic tactics which are so fundamental to the squared-error and graph–theoretic techniques, so this method has limited applicability. Section 3.1 compares the intrinsic natures of the programs with the admissibility criteria suggested by Fisher and Van Ness.[28]

A third method of comparison is to establish a measure of similarity between pairs of clustering programs based on performance when the same set of data is presented to all methods.[29] Section 3.2 uses the procedure suggested by Anderberg,[1] which is to cluster the clustering programs themselves based on some measure of similarity.

A unique feature of the comparisons completed in this section is their objectivity. Most papers on clustering are trying to sell a particular method so the comparison with other methods is devoted to highlighting the characteristics of the clustering algorithm being proposed.[30] In this paper, a well-rounded comparison is provided, concentrating on those aspects which are essential to the user.

The difficulties inherent in the comparison problem can be appreciated somewhat by considering the set of two-feature patterns in Fig. 4. The clusterings achieved by the programs studied in this paper are shown. Since all edge weights in the minimum spanning tree are unity, the dendrogram from the single-link hierarchical clustering program is trivial—all patterns are placed in the same cluster at the first step. Which of the programs is best? What is the "true" structure?

3.1. *Admissibility criteria*

Admissibility criteria provide one of the only ways of comparing clustering techniques themselves. The idea is to propose a property, P, and call a clustering technique P-admissible if it satisfies P. The property P should be something that is satisfied by any "reasonable" technique. For details, see Fisher and Van Ness[28] and Rubin.[31]

The first set of properties relates to the manner in which clusters are formed.

(1) A clustering technique produces an *image-admissible* clustering if no other clustering exists having the same number of clusters and the same number of patterns per cluster that is "uniformly better" than the one produced by the technique.

(2) *Convex admissibility* means that the convex hulls of the clusters do not intersect.

(3) *Connected admissibility* is less restrictive than convex admissibility and was defined[28] only for the case when the pattern space has two dimensions. However, it can be extended to the general case as follows. Consider the MST (minimum spanning tree), as in the MSTCLUS program. Let $(C_1, C_2, \ldots, C_K)$ be a clustering and let L_i be the smallest of the MST needed to connect all nodes in C_i. The clustering is connected admissible if it always generates this type of clustering.

The next two properties reflect the structure of the data.

(4) *Exact tree:* The data satisfy the ultrametric inequality.[15] This implies that the matrix of distances between all pairs of patterns can be reproduced from the dendrograms generated by the single-link and complete-link methods of hierarchical clustering. These dendrograms are identical in this case. An exact-tree admissible technique will find this dendrogram.

(5) *K-group:* A K-cluster clustering exists in which all within-cluster distances are smaller than all between-cluster distances. This is the case of well separated, tight clusters. A K-group admissible technique will find these clusters.

The final four properties test the sensitivity of the clustering technique to changes which do not alter the essential structure of the data.

(6) *Point proportion:* Duplication of one or more patterns any number of times does not change the boundaries of the clusters when a clustering technique has this property.

(7) *Cluster proportion:* Duplication of all patterns in one or more clusters any number of times does not change the cluster boundaries.

(8) *Cluster omission:* Removal of an entire cluster does not change the clustering on the remaining patterns when the technique is reapplied.

(9) *Monotone:* A monotone transformation applied to the dissimilarity measure does not change the clustering.

Table 1 lists the admissibility properties for all the clustering techniques. The four squared-error programs are grouped because they are all representative

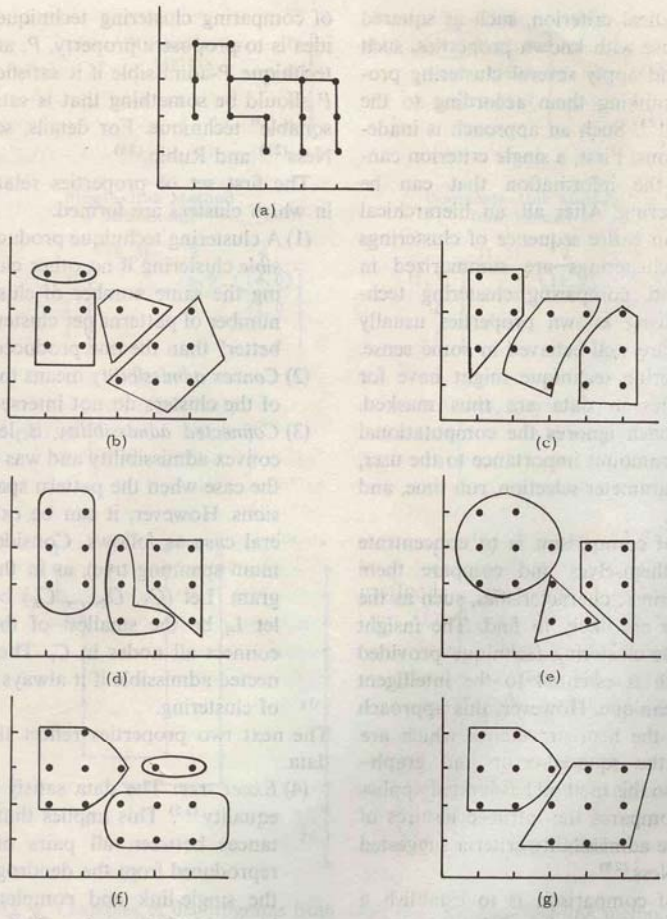


Fig. 4. Several clusterings of fifteen patterns in two dimensions: (a) fifteen patterns with minimum spanning tree shown (b) clusters from FORGY (c) clusters from ISODATA (d) clusters from WISH (e) clusters from CLUSTER (f) clusters from complete-link hierarchical program (g) clusters from JP.

of the same technique. Program SINLNK is grouped with program MSTCLUS because the clustering selected from SINLNK is obtained by cutting the dendrogram, which is equivalent to cutting the MST.

The computationally unfeasible minimum-squared-error technique, the ideal for all the squared-error programs, is included for comparison only.

3.2 *Empirical study*

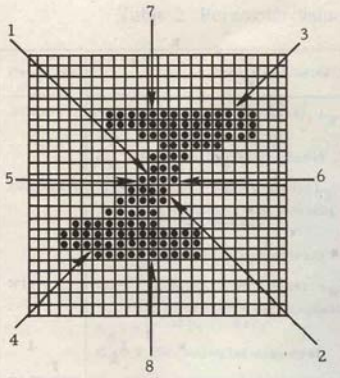
The eight clustering programs were applied to the particular pattern matrix, derived from the Munson handprinted Fortran character set,* which consists of

*Available from the Computer Society, Institute of Electrical and Electronic Engineers.

Table 1. Admissibility properties

Program \ Condition	Image	Convex	Connected	Exact tree	k-group	Point proportion	Cluster proport.	Monotone	Cluster omission
FORGY, WISH, ISODATA, CLUSTER	NO	YES	YES	*	NO	NO	NO	NO	*
Minimum-squared error (K fixed)	YES	YES	YES	*	YES	NO	NO	NO	YES
SINLNK, MSTCLUS	YES	NO	YES	YES	YES	YES	YES	YES	YES
COMLNK	YES	NO	NO	YES	YES	YES	YES	YES	YES
JP	NO	NO	YES	*	YES	NO	NO	YES	*

* Not applicable



Feature vector (pattern): (11, 11, 5, 6, 10, 10, 5, 5)

Fig. 5. Example of feature definitions from IMOX data base.

binary-coded (24 × 24) handwritten characters from several authors. Each author in the Munson data base prepared three alphabets of 46 characters. The pattern matrix generated for this study, called the IMOX data base, uses all three alphabets from the first 16 authors. Four characters were selected from each alphabet, namely I, M, O, and X, for a total of 192 characters. Each character is represented by an eight-dimensional pattern. The features are the number of squares from the perimeter to the character, as demonstrated in Fig. 5. Similar features have been used elsewhere.[30,32] This pattern matrix has reasonable size (192 × 8), can be reproduced by anyone having a copy of the Munson tape, and presents a challenge to a clustering program. As will be seen, the patterns are not well clustered according to category, but are reasonably clustered.

Two two-dimensional representations of the eight-dimensional IMOX pattern matrix are shown in Figs. 6 and 7. Both are linear projections and demonstrate that the patterns do not cluster in a trivial manner. The patterns are identified only according to category (pattern class). Figure 6 shows the two best Karhunen–Loéve coordinates[8] with 57.9% of the variance retained. Figure 7 shows the two dimensions from the three-dimensional discriminant plane in which the scatter ratio is minimized.[9,10]

3.2.1 *Individual clustering programs.* Tables 2 and 3 summarize the results of applying the eight clustering programs to the IMOX data base. All computation was performed on the CDC 6500 computer system at Michigan State University. All the nonhierarchical programs (except CLUSTER) were run

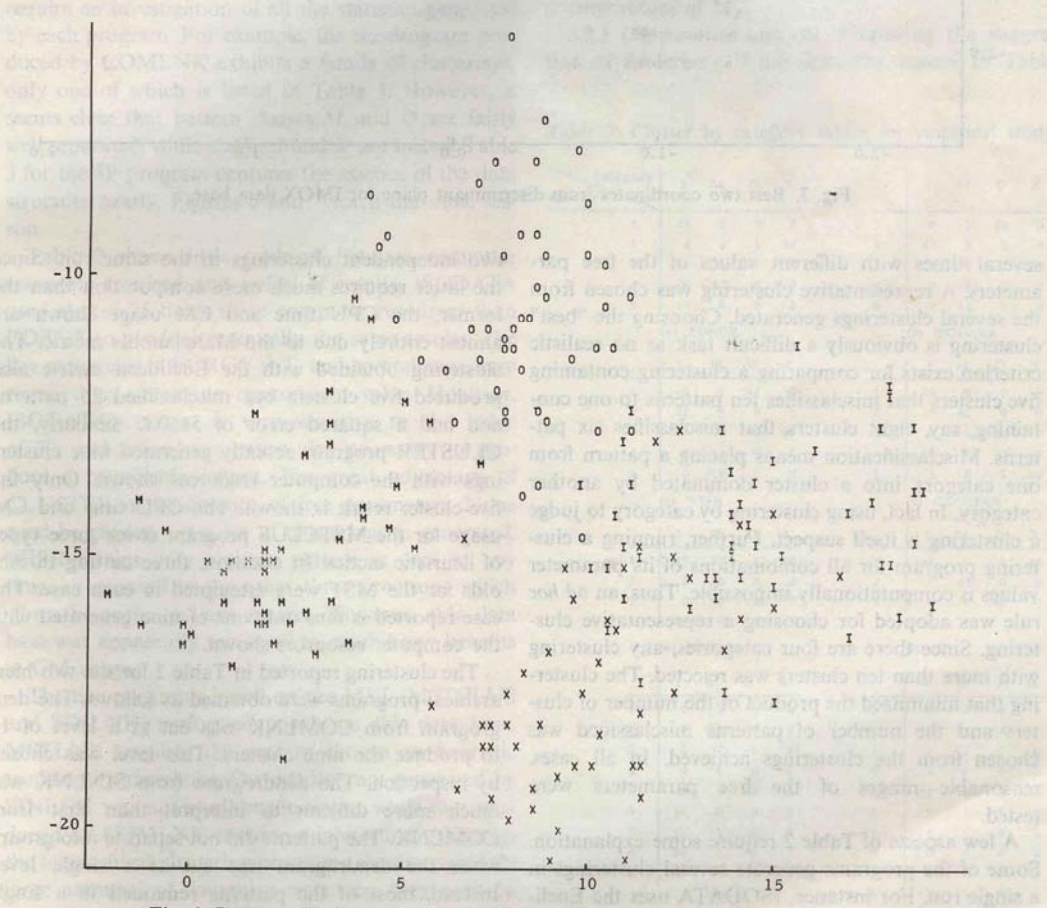


Fig. 6. Best two Karhunen–Loéve coordinates of (unnormalized) IMOX data base.

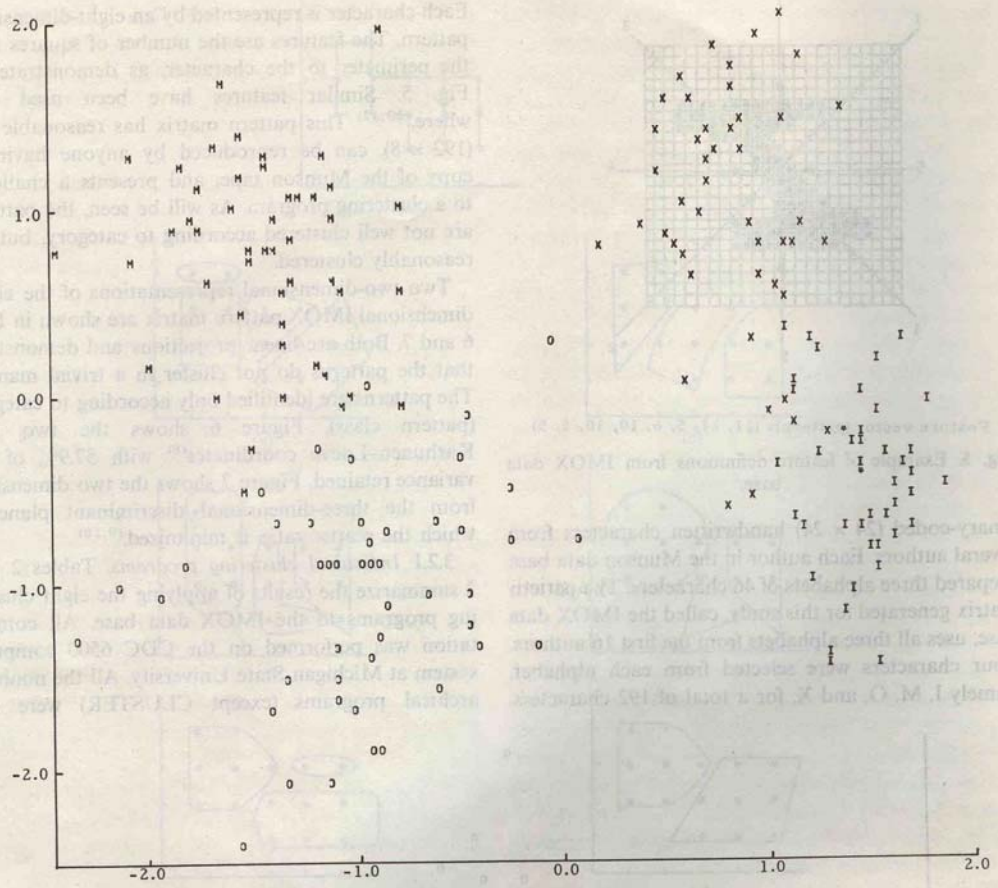


Fig. 7. Best two coordinates from discriminant plane for IMOX data base.

several times with different values of the free parameters. A representative clustering was chosen from the several clusterings generated. Choosing the "best" clustering is obviously a difficult task as no realistic criterion exists for comparing a clustering containing five clusters that misclassifies ten patterns to one containing, say, eight clusters that misclassifies six patterns. Misclassification means placing a pattern from one category into a cluster dominated by another category. In fact, using clustering by category to judge a clustering is itself suspect. Further, running a clustering program for all combinations of its parameter values is computationally impossible. Thus, an *ad hoc* rule was adopted for choosing a representative clustering. Since there are four categories, any clustering with more than ten clusters was rejected. The clustering that minimized the product of the number of clusters and the number of patterns misclassified was chosen from the clusterings achieved. In all cases, reasonable ranges of the free parameters were tested.

A few aspects of Table 2 require some explanation. Some of the programs generate several clusterings in a single run. For instance, ISODATA uses the Euclidean metric and the Mahalanobis metric to produce two independent clusterings in the same run. Since the latter requires much more computation than the former, the CPU time and CM usage shown are almost entirely due to the Mahalanobis metric. The clustering obtained with the Euclidean metric also produced five clusters but misclassified 36 patterns and had a squared error of 5150.1. Similarly, the CLUSTER program actually generated nine clusterings with the computer resources shown. Only the five-cluster result is shown. The CPU time and CM usage for the MSTCLUS program cover three types of heuristic tactics. In addition, three cutting thresholds for the MST were attempted in each case. The case reported is thus only one of nine generated with the computer resources shown.

The clustering reported in Table 2 for the two hierarchical programs were obtained as follows. The dendrogram from COMLNK was cut at a level of 15 to produce the nine clusters. This level was chosen by inspection. The dendrogram from SINLNK was much more difficult to interpret than that from COMLNK. The patterns did not separate into groups when the dendrogram was cut at a single level. Instead, most of the patterns remained in a single cluster. The SINLNK and COMLNK clusterings

Table 2. Parameter values and computer resources used for empirical study

Program	Free Parameter Values	CPU time (sec.)	CM usage (word-hr.)	Squared error	No. of clusters	No. misclassified
FORGY	$MIN_F = 2$ (min. cluster size); $L_F = 10$ (max. iterations/loop) $T_F = 0.15$ (threshold) $K_F = 4$ (no. of initial clusters)	4.91	0.74	4942.8	5	21
ISODATA	$MIN_I = 1$ (min. cluster size); $L_I = 50$ (max. iterations/loop) $T_{I1} = 0.25$ (splitting and lumping thresholds) $T_{I2} = 0.75$ $K_I = 4$ (no. of clusters desired)	106.90	11.18	5149.6	5	18
WISH	$MIN_W = 2$ (min. cluster size); $r_W = 2.0$ (parameter used in selecting initial cluster centers) $T_W = 0.3$ (thresholds for disposing of patterns) $\theta_W = 8.0$ $K_W = 2$ (no. of initial clusters)	33.97	3.33	5046.0	5	27
CLUSTER	$K_C = 10$ (max. no. of clusters generated)	27.48	4.07	4942.8	5	22
SINLNK	None	128.27	27.89	-	4	60
COMLNK	None	129.14	27.90	-	9	32
MSTCLUS	$r_M = 2$ (point density heuristics)	50.31	7.63	-	10	96
JP	$M_J = 21$ (size of near-neighbor table) MATCH = 5 (no. of matches required)	25.31	1.67	-	3	49

reported in Tables 2 and 3 were obtained by a strictly *ad hoc* procedure.

3.2.2 *Program comparison.* Tables 2 and 3 provide some interesting comparisons of the eight programs. Drawing conclusions about the data base itself would require an investigation of all the statistics generated by each program. For example, the dendrogram produced by COMLNK exhibits a family of clusterings, only one of which is listed in Table 3. However, it seems clear that pattern classes *M* and *O* are fairly well separated, while classes *I* and *X* are mixed. Table 3 for the JP program captures the essence of the data structure neatly. Figures 6 and 7 teach the same lesson.

Table 2 shows little trade-off between computer resources required and utility of results. ISODATA used almost 25 times more computer resources than FORGY to produce essentially the same results. The key parameter in FORGY is T_F and several runs were required to obtain an appropriate value. However, ISODATA's output is very sensitive to the value of T_{I2} and several runs were also necessary to produce a reasonable output. The great advantage of CLUSTER is the absence of free parameters. None need be chosen, yet a set of clusterings is generated. WISH retains only a single pattern in core at any time, as opposed to the other three programs which store the entire pattern matrix. However, this data base was apparently too small to exhibit any benefits in CM usage.

The two programs based on the MST (MSTCLUS and SINLNK) performed poorly on this data base. MSTLNK requires the user to choose an heuristic tactic and a threshold. The dendrogram generated by SINLNK was not at all encouraging, while that produced by COMLNK was most interesting. Although the hierarchical clustering programs require relatively large amounts of computer resources, they need be run only once and provide many clusterings from the dendrogram. The CPU time shown for JP does not include preparation of the near-neighbor table (which took 19.9 sec). A series of runs were made for different values of MATCH, with the most noteworthy being reported. This program also needs to be run with several values of M_J.

3.2.3 *Comparative analysis.* Following the suggestion of Anderberg,[1] the similarity matrix in Table

Table 3. Cluster by category tables for empirical study.

Cluster \ Category	I	M	O	X
1	25	0	0	8
2	4	0	1	38
3	0	6	47	0
4	0	42	0	0
5	19	0	0	2

(a) FORGY

	I	M	O	X
1	0	2	46	0
2	43	0	0	9
3	5	0	1	39
4	0	32	1	0
5	0	14	0	0

(b) ISODATA

	I	M	O	X
1	0	1	47	1
2	31	0	0	7
3	17	0	1	40
4	0	33	0	0
5	0	14	0	0

(c) WISH

	I	M	O	X
1	25	0	0	8
2	0	42	0	0
3	0	6	47	0
4	18	0	0	2
5	5	0	1	38

(d) CLUSTER

	I	M	O	X
1	19	19	0	11
2	0	28	0	0
3	0	1	48	0
4	29	0	0	37

(e) SINLNK

	I	M	O	X
1	48	0	0	48
2	0	47	0	0
3	0	1	48	0

(f) JP

	I	M	O	X
1	0	5	0	0
2	0	21	0	0
3	0	7	0	0
4	0	6	0	2
5	0	9	48	0
6	16	0	0	2
7	10	0	0	5
8	16	0	0	8
9	6	0	0	31

(g) COMLNK

	I	M	O	X
1	7	0	0	0
2	22	37	48	20
3	1	0	0	0
4	1	0	0	0
5	17	0	0	24
6	0	1	0	0
7	0	9	0	0
8	0	1	0	0
9	0	0	0	2
10	0	0	0	2

(h) MSTCLUS

Table 4. Rand's measure of similarity for the clusterings reported in Tables 2 and 3

	2	3	4	5	6	7	8
1	0.9966	0.9035	0.8102	0.8968	0.5979	0.8012	0.8613
2	-	0.9015	0.8091	0.9000	0.5972	0.7998	0.8633
3	-	-	0.8432	0.9127	0.5772	0.8177	0.8556
4	-	-	-	0.8364	0.5307	0.8319	0.7513
5	-	-	-	-	0.5629	0.8114	0.8593
6	-	-	-	-	-	0.5617	0.6374
7	-	-	-	-	-	-	0.8186

Key: 1: CLUSTER 5: ISODATA
2: FORGY 6: MSTCLUS
3: WISH 7: SINLNK
4: JP 8: COMLNK

4 for the eight clustering programs was computed for the clusterings in Table 3. See the Appendix for details. The single-link and complete-link dendrograms for this similarity matrix are shown in Fig. 8. Kruskal's multidimensional scaling[5] produced the one-dimensional configuration in Fig. 9. The stress was less than 1%, so Fig. 9 is a valid representation of the performance of the clustering programs on the IMOX data.

Figures 8 and 9 both summarize the comparative analysis by providing pictures of the similarity matrix in Table 4. All eight clusterings provide views of the data. A user would like the clusterings to provide as different views as possible. It might seem that one could cover the gamut of possible clusterings by using a squared-error program, an hierarchical program, and a graph–theoretic program. Figure 9 provides little evidence for this belief.

The four squared-error programs produced relatively similar clusterings, as expected. However, the COMLNK clustering was similar to all of them and, from Fig. 9, can be considered as an average of the squared-error clusterings. The two least similar clusterings, MSTCLUS and JP, are both graph–theoretic clusterings. Finally, the MSTCLUS clustering is significantly different from all the rest, as seen by the scale in Fig. 9. This evidence indicates that employing a new clustering method will not necessarily produce a unique clustering. Indeed, Fig. 9 suggests that MSTCLUS, COMLNK, and JP are sufficient to provide several alternative hypotheses about the structure of this data base.

4. CONCLUSIONS

This paper takes the user's point of view so the *conclusions will be directed toward potential users* of clustering programs. A user must remember that a clustering program is a tool for discovery, not an end in itself. A cluster analysis is really a preprocessing step that should generate ideas and help the user form hypotheses. A cluster analysis should be supplemented by other descriptive techniques, such as the two-domensional projections in Figs. 6 and 7. The utility of a cluster analysis is more in the questions raised than in the questions answered.

The descriptive goals of a clustering method or program make the intepretation of the results subjective. Someone involved in the subject matter of the data base is usually able to interpret the results of a cluster analysis more meaningfully than the clustering professional. This subjective quality of the results make it essential that the user try several clustering programs, not merely one that happens to be available. Looking for an "optimum" clustering method is contrary to the nature of the problem. The intelligent user must be aware of differences among clustering methods and

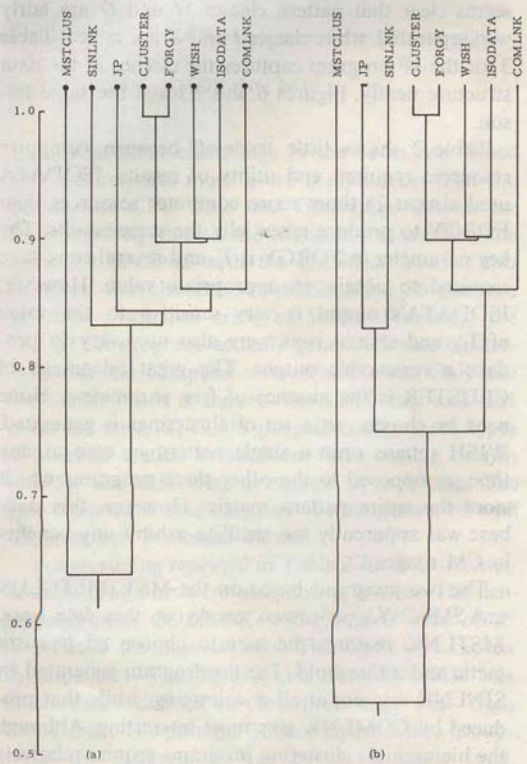


Fig. 8. Dendrograms for comparative analysis of eight programs: (a) single link (b) complete link.

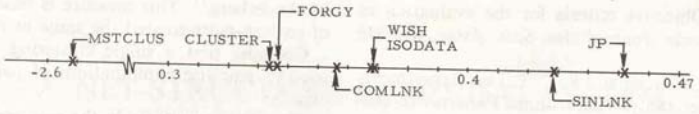


Fig. 9. Multidimensional scaling representation of Table 4.

of mathematical and statistical links to the methodology.

What guidelines can be established for helping the user choose a clustering method? The empirical study in Section 3 implies that no such guidelines exist, other than choosing the type of output. However, several computational guidelines can be drawn. The computer facilities available and the size of the data base to be clustered could be the crucial factors in selecting a clustering program. For example, the SINLNK and COMLNK programs store, at least temporarily, two copies of the entire (upper triangular) similarity matrix. All the squared-error programs except WISH store the entire pattern matrix. Such computational realities can severely limit a user's options. Hall *et al.*[3] suggest some approximate procedures and Mucciardi and Gose[14] propose a noniterative, quick-look clustering procedure for very large data bases.

When the data base is small and the computer facilities ample, as in the empirical study, some theoretically interesting methods are computationally unattractive, as revealed in Table 2. For a given cost, it seems much better to run a simple program, such as FORGY, several times than to run a sophisticated program, such as ISODATA with a Mahalanobis metric, once. The CLUSTER program is attractive because it has no free parameters and produces several clusterings. In the experience of the authors, the dendrogram from the COMLNK program is a popular choice among naïve users of clustering programs. In fact, the minimum spanning tree itself, on which MSTCLUS and SINLNK are based, appears more valuable in summarizing data than do the clusterings based on heuristic notions in MSTCLUS. Hall *et al.*[3] make a similar observation.

It appears that enough clustering algorithms known to uncover specific data structures are available, even though few actual structures can be identified. Future research in clustering methodology should aim at providing a rational basis for comparing clustering methods. The works of Ling,[33] Hubert,[21,34] and Maronna[35] move in this direction Links to well-understood mathematical and statistical methods also need to be forged.

REFERENCES

1. M. R. Anderberg, *Cluster Analysis for Applications*, Academic Press, NY (1973).
2. B. Everitt, *Cluster Analysis*, Wiley, NY (1974).
3. D. J. Hall, R. O. Duda, D. A. Huffman and D. E. Wolf, Development of new pattern recognition methods, Aerospace Research Laboratories Report, 73-0153, Wright Patterson Air Force Base, Ohio (November 1973). Also available as AD-772614.
4. J. W. Sammon, Jr., A nonlinear mapping for data structure analysis, *IEEE Trans. Computers* **18,** 401 (May 1969).
5. J. B. Kruskal, Nonmetric multidimensional scaling: a numerical method, *Psychometrika* **29,** 115 (June 1964).
6. J. B. Kruskal, Comments on a nonlinear mapping for data structure analysis, *IEEE Trans. Computers* **20,** 1614 (December 1971).
7. C. K. Chen and H. C. Andrews, Nonlinear intrinsic dimensionality computations, *IEEE Trans. Computers* **23,** 178 (February 1974).
8. K. Fukunaga, *Introduction to Statistical Pattern Recognition*, Academic Press, NY (1972).
9. S. S. Wilks, *Mathematical Statistics*, Wiley, NY (1963).
10. W. W. Cooley and P. R. Lohnes, *Multivariate Data Analysis*, Wiley, NY (1971).
11. R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, Wiley, NY (1973).
12. H. P. Friedman and J. Rubin, On some invariant criteria for grouping data, *Journal Am. Stat. Assoc.* **62,** 1159 (December 1967).
13. G. Sebestyen and J. Edie, An algorithm for nonparametric pattern recognition, *IEEE Trans. Electronic Computers* **15,** 908 (December 1966).
14. A. N. Mucciardi and E. E. Gose, An automatic clustering algorithm and its properties in high-dimensional spaces, *IEEE Trans. Syst. Man, Cybern.* **2,** 247 (April 1972).
15. S. C. Johnson, Hierarchical clustering schemes, *Psychometrika* **32,** 241 (1967).
16. N. Jardine and R. Sibson, *Mathematical Taxonomy*, Wiley, NY (1971).
17. P. H. A. Sneath and R. R. Sokal, *Numerical Taxonomy*, Freeman, San Francisco (1973).
18. J. C. Gower and G. J. S. Ross, Minimum spanning tree and single-linkage cluster analysis, *Appl. Stat.* **18,** 54 (1969).
19. R. Sibson, SLINK—optimally efficient algorithm for single-link cluster method, *Comp. J.* **16,** 30 (1973).
20. F. J. Rohlf, Hierarchical clustering using minimum spanning tree, *Comput. J.* **16,** 93 (1973).
21. L. Hubert, Approximate evaluation technique for the single-link and complete-link hierarchical clustering procedure, *Journal Am. Stat. Assoc.* **9,** 968 (1974).
22. C. T. Zahn, Graph–theoretic methods for detecting and describing gestalt clusters, *IEEE Trans. Computers* **20,** 68 (January 1971).
23. A. Jarvis and E. A. Patrick, Clustering using a similarity measure based on shared near-neighbors, *IEEE Trans. Computers* **22,** 1025 (November 1973).
24. R. A. Fisher, The use of multiple measurements in taxonomic problems, *Ann. Eugenics* **3,** 179 (1936).
25. J. J. Freeman, Experiments in discrimination and classification, *Pattern Recognition* **1,** 207 (March 1969).
26. J. C. Gower, A comparison of some methods of cluster analysis, *Biometrics* **23,** 623 (1967).
27. W. E. Wright, A formalization of cluster analysis, *Pattern Recognition* **5,** 273 (1973).
28. L. Fisher and J. W. Van Ness, Admissible clustering procedures, *Biometrika* **58,** 91 (1971).

29. W. M. Rand, Objective criteria for the evaluation of clustering methods, *Journal Am. Stat. Assoc.* **66,** 846 (1971).
30. J. R. Slagle, C. L. Chang and R. C. T. Lee, Experiments with some cluster analysis algorithms, *Pattern Recognition* **6,** 181 (1974).
31. J. Rubin, Optimal classification into groups: an approach for solving the taxonomy problems, *J. Theoret. Biol.* **15,** 103 (1967).
32. K. S. Fu, *Sequential Methods in Pattern Recognition and Machine Learning*, Academic Press, NY (1968).
33. R. F. Ling, Probability theory of cluster analysis, *J. Am. Stat. Assoc.* **68,** 159 (1973).
34. L. J. Hubert, Some applications of graph theory for clustering, *Psychometrika* **39,** 283 (1974).
35. R. Maronna and P. M. Jacovkis, Multivariate clustering procedures with variable metrics, *Biometrics* **30,** 499 (1974).

APPENDIX

Measure of similarity between clusterings

The analysis in Section 3.2 of the clusterings produced by the eight programs uses the measure of similarity between clusterings proposed by Rand[29] and discussed by Anderberg.[1] This measure is based on the percentage of pattern-pairs treated the same in two clusterings.

Consider first a single clustering. Assign each of the $n(n\text{-}1)/2$ pairwise combinations of patterns to one of two classes:

Class 0: the patterns in the pair are in the same cluster;
Class 1: the patterns in the pair are in different clusters.

Given two clusterings, a contingency table is constructed having the form shown below.

Clustering A / Clustering B	1	0
1	a	b
0	c	d

Thus, c is the number of pattern pairs which are placed in different clusters by clustering A, but in the same cluster by clustering B. Rand's measure of similarity between the two clusterings is:

$$S(A, B) = \frac{a + d}{a + b + c + d}.$$

The larger $S(A, B)$ is, the more similar the two clusterings are.

About the Author—RICHARD C. DUBES was born in Chicago, Illinois in 1934. He received the B.S. degree from the University of Illinois in 1956, the M.S. degre from Michigan State University in 1959 and the Ph.D. degree from Michigan State University in 1962, all in electrical engineering. In 1956 and 1957, he was a member of the technical staff of the Hughes Aircraft Company, Culver City, California. From 1957 through 1968, he served as graduate assistant, research assistant, assistant professor, and associate professor in the Electrical Engineering Department at Michigan State University. In 1969, he joined the Computer Science Department at Michigan State University and became Professor in 1970. He is the author of *The Theory of Applied Probability* (Prentice-Hall, 1968) and several technical papers and reports.

Dr. Dubes has served as a consultant to the Lear–Siegler Corp., Grand Rapids, Michigan and the J. M. Richards Laboratory, Detroit, Michigan. His areas of technical interest include pattern recognition, clustering, decision theory, and application of data analysis methods to the medical area. He is a member of the Institute of Electrical and Electronic Engineers, the Pattern Recognition Society, and Sigma Xi.

About the Author—ANIL K. JAIN was born in Basti, India, on 5 August, 1948. He received the B.Tech. degree with distinction from Indian Institute of Technology, Kanpur in 1969, and the M.S. and Ph.D. degrees in electrical engineering from Ohio State University, Columbus, in 1970 and 1973, respectively. He was recipient of the National Merit Scholarship in India.

From 1971 to 1972 he was a Research Associate at the Communications and Control Systems Laboratory, Ohio State University, working on Dimensionality and Sample Size problems in statistical pattern recognition. Then, from 1972 to 1974, he was an Assistant Professor in the Computer Science section at Wayne State University, Detroit. He is currently an Assistant Professor in the Computer Science Department at Michigan State Universtiy, doing research in pattern recognition theory and applications.

Dr. Jain is member of the Association for Computing Machinery and the Institute of Electrical and Electronic Engineers.