CHAPTER 1 - ALGORITHMS AND APPLICATIONS

1.1 INTRODUCTION

Automatic classification algorithms are used in widely different fields in order to provide a description or a reduction of data. The data items to be classified are generally quantitatively characterised. If not, in the case of nominal variables for example, they may be incorporated into this framework by the use of a suitable coding (such as 1 = possession of a property, 0 = otherwise).

Any clustering algorithm will attempt to determine the inherent or natural groupings in the data. More concretely, motivation for clustering may be categorized under four headings, as follows.

(a) Data analysis

Here, the given data is to be analysed in order to reveal its fundamental features. The significant interrelationships present in the data are sought. This is the multivariate statistical use of clustering, and the validity problem (i.e. the validation of clusters of data items produced by an algorithm) is widely seen as a major current difficulty.

(b) User convenience

A synoptic classification is to be obtained which will present a useful decomposition of the data. This may be a first step towards subsequent statistical analysis, or it may simply entail the provision of a user-friendly interface to an information system. The emphasis is on heuristics for summarizing information. Appraisal of clustering algorithms used for this purpose can include : algorithmic efficiency, flexibility for input data, clarity of output presentation, and ease of use.

(c) Storage and retrieval

We are here concerned with improving access speeds by providing better routing strategies to stored information. The effectiveness of the clustering is measured by time and space efficiency, and by external criteria (related, for example, to the amount of relevant material retrieved).

(d) Machine vision

The distinguishing of point patterns, or the processing of digital image data, is often assessed visually, although computational efficiency is important also.

Applications of clustering therefore embrace many far-flung fields. The range of algorithms which have been proposed (for the most part, since the early 1960s with the advent of computing power on a wide scale) has been correspondingly large.

In general, it will be supposed that the given data consists of objects (or items or individuals) and variables (or attributes). The interrelationship values may be numerically specified, and a data array or matrix is presented to the algorithm. This does not, however, preclude suitable, alternative forms of storage for such data - for example, in the case of very sparse document-term matrices in information retrieval; or of pixel readings, arranged as images, in image processing.

The emphasis in the following sections is on algorithms which are simply described and which are immediately applicable.

Section 1.2 describes two algorithms which do no more than re-arrange the rows and columns of an array. By "pulling together" large values they succeed in providing an improved visual presentation of information.

These algorithms are unusual in that they do not appeal directly to the notion of distance or similarity. A distance function may be defined on pairs of row (or column) vectors. A brief introduction to requirements in this area is given in section 1.3.

Most published work in cluster analysis involves the use of either of two classes of clustering algorithm : a hierarchical or a non-hierarchical (or partitioning) algorithm. Sections 1.4 and 1.5 provide introductions to these two classes of algorithm. These sections are centred on a very widely-employed hierarchical clustering technique - the single link method - and on two variants (connected components and the minimal spanning tree). A major reason for the wide interest in these approaches is that they can be implemented efficiently. For large data sets (in astronomy and in information retrieval, for instance), these have been considered until recently to provide the only computationally feasible hierarchical clustering strategies.

1.2 MATRIX REORDERING TECHNIQUES

A set of numbers is often easier for the human to interpret when sorted; and the automatic search for the presence of a particular value is likewise facilitated when the set of values is ordered.

The sorting of the rows and columns of an array, according to some definition of row/column weights, can reveal structure in the array. Seriation, a problem arising in the analysis of questionnaires, and in the analysis of archaeological data, affords a particularly clear example of the pattern sought in the array (Fig. 1.1). In this case, the permutation of the rows and then of the columns so that the row (column) totals are in non-increasing order, will find this pattern. For more general real-valued matrix entries, some other row (column) weights may be defined which adequately discriminate as to where large and small array values are positioned. The objective in the following algorithm is to place as many large array values on the diagonal as possible. Let a_{ij} be the value of row i and column j; $a_i = \Sigma_j a_{ij}$ = the sum of row i elements, and similarly a_j is the sum of column j elements.

Algorithm A. Iterative matrix reordering

Step 1 Calculate the weight of each row i :

$$w_i = \Sigma_j (a_{ij}/a_i) \cdot j$$

Step 2 Reorder rows in decreasing order of weights.

Step 3 Calculate the weight of each column (defined analogously to the row weights).

Step 4 Reorder columns in decreasing order of weights.

Step 5 Return to Step 1, until either no reordering of rows or columns was, in fact, necessary; or a maximum number of iterations has been carried out.

The role of the denominators in the expressions for weights (a_i and a_j for rows and columns, respectively), which are not governed by the summation, is to correct or normalize for large values in the row (column) which would otherwise overly influence the resulting weight. The permuting of columns in Step 4 will ordinarily interfere with the ordering of rows, carried out in Step 2. Therefore a recalculation of new row weights, and subsequent reordering, is called for. The permuting of columns is then required again. If either reordering introduces no change, at any stage of the iteration, then convergence has

Q1 Have you had primary education?

Q2 Have you had second level education?

Q3 Have you had a primary (bachelor) degree or equivalent?

Q4 Have you had a higher university degree?

	Q1	Q2	Q3	Q4
S1	1	1	1	1
S2	1	1	1	0
S3	1	1	0	0
S4	1	0	0	0
S5	0	0	0	0

1: Yes
0: No

With few exceptions, test subjects will display one of the five response patterns shown.

	Questions			
	1	1	1	1
	1	1	1	0
	0	0	0	0
	0	1	0	0
Subject	0	1	1	0
responses	0	1	0	0
	1	1	1	1
	0	1	1	0
	0	1	0	0
	0	0	0	0
	1	1	1	1

Fig. 1.1 - Reordering on the basis of row/column totals would reveal the types of test subject, and also the most suitable ordering of questions.

been attained. The algorithm can loop infinitely between a number of solution states, and one of these may be chosen as adequate. The initially given ordering of rows and columns can also **affect** the calculation and so the algorithm should preferably be executed with a number of different permutations of rows and columns. An example of Algorithm A is shown in Fig. 1.2.

The simultaneous reordering of rows and columns is of greatest interest when both rows and columns depend on the same underlying variable. For example, both archaeological artifacts and their properties are primarily related to chronological ordering in time; or, both examination questions and examinees are related to the inherent difficulty of the examined material - the examinees will be ordered by degree of comprehension, while the examination questions will be ordered by degree of difficulty. If a concentration of large-valued elements on the diagonal results from applying Algorithm A, then almost certainly it will indicate a useful, common interpretation of the rows and columns. On the other hand, if the solution provided by the algorithm does not achieve this concentration of large elements, then the perspective which is being applied - the model of one-dimensionality of rows and of columns - is over-simplistic for the data.

Other studies where Algorithm A has been successfully employed include its use for the more informative presentation of tabular data : Deutsch and Martin (1971) use it to arrange a table of countries' voting record at the United Nations, so that international power blocks and regional areas appear clearly.

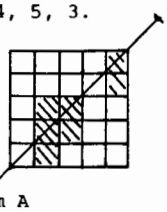
3	0	0	2	1
0	0	3	1	0
2	0	1	3	1
3	1	0	1	0
1	0	3	0	1

Given array

0	0	1	0	3
0	1	0	1	3
0	2	3	1	1
0	3	2	1	0
1	3	1	0	0

Array, as reordered by Algorithm A

Final ordering of rows 2, 5, 3, 1, 4.
Final ordering of columns 2, 1, 4, 5, 3.



Values $\geq$ 2 are shaded.

Fig. 1.2 - Example of Algorithm A

Egan (1977) assessed 100 Irish children aged between 12 and 17 on school essays on the topic of "Ireland : What it Means to Me". Each sentence in the essays contributed an increment to one possible attribute among 38 (physical environment, places, ..., industry and technology, ..., historical events, ..., art, language, Irishness). Reordering the resulting 100 x 38 array led to a clear ordering of the 38 content-items. A subdivision of the children into younger and older was also brought about, so that the ordering of the 38 content-items was related to an underlying scale of affective develoment. Among the points made by Egan for the use of Algorithm A were the following : it captures a good deal of a known underlying order; it provides a framework for intuition and forces greater clarity; it is relatively objective, is fast, and can be implemented in large-scale research.

Algorithm A attempts to reposition large array values along the diagonal, while preserving intact the initially-given array. It would frequently be of greater interest to simply attempt to clump or cluster large values together. A criterion of clumpiness is the product of each array element with its four adjacent elements, summed over all array values :

$$\sum_i \sum_j a_{ij}(a_{i-1,j} + a_{i+1,j} + a_{i,j-1} + a_{i,j+1})$$

with, by convention

$$a_{oj} = a_{n+1,j} = a_{io} = a_{m+1,0} = 0 \quad (n,m : \text{numbers of rows, columns})$$

An optimal value of this expression is sought, where the only permissible operations on the array are row and column reorderings. However, in reordering rows, the final two terms in the above expression will be inactive (these only concern values within the row considered) and so the criterion to be optimized is

$$\sum_i \sum_j a_{ij}(a_{i-1,j} + a_{i+1,j}) \qquad (1)$$

Similarly, in reordering columns, the criterion to be optimized becomes :

$$\sum_i \sum_j a_{ij}(a_{i,j-1} + a_{i,j+1}) \qquad (2)$$

The following algorithm will provide a good, albeit suboptimal, solution to this problem, beginning with the reordering of rows.

Algorithm B. Non-iterative matrix reordering

Step 1 Select a row i arbitrarily.

Step 2 For each remaining row in turn, determine the value of expression (1) given by placing it to the left or right of any row already positioned; place the row which gives the greatest value of expression (1) in its best position.

Step 3 When all rows have been placed, repeat Steps 1 and 2 for columns (using expression (2)).

Initially, in Step 2, we will seek the closest row (employing expression (1)) to the row chosen in Step 1. At any subsequent execution of Step 2, a new row will be a candidate for placement in k+1 locations, where there are k rows already placed.

Since the reordering of columns does not **affect** the prior ordering of rows (i.e. the value obtained for expression (1) is unchanged), Algorithm B is non-iterative. If the array is symmetric, the reordering of columns will necessarily be the same as the reordering found for the rows, and so only Steps 1 and 2 will be required.

Finally, Algorithm B is dependent on the initial choice of row (in Step 1) and column. A few different start choices will verify the robustness of the solutions obtained. An example of the implementation of Algorithm B is shown in Fig. 1.3.

Algorithm B is similar to Algorithm A in that it carries out a straightforward rearranging of elements of an array, but is perhaps more widely applicable. McCormick et al. (1972) discuss a planning example, where interrelationships of a set of 27 airport subsystems (passenger check-in, baggage check-in, short-period parking areas, long-term parking, refuse removal, ...) are subjectively characterized on a 0-3 scale. Algorithm B then provides an exploratory set of clusters. Another example uses an interrelation table crossing marketing applications with various O.R. and statistical techniques, and illustrates how Algorithm B gives a more orderly presentation of tabular information to a user. Finally, March (1983) reports on the application of Algorithm B to the problem of structuring records in a database. A similarity measure between data items is used which takes account of different accessing practices of a user population. Efficiency and retrieval is enhanced if items showing similar access characteristics (and which will, with a high likelihood, be required together) are physically stored together.

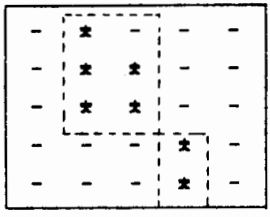
	1	2	3	4	5
1	3	0	0	2	1
2	0	0	3	1	0
3	2	0	1	3	1
4	3	1	0	1	0
5	1	0	3	0	1

Given array

	5	1	4	3	2
4	0	3	1	0	1
1	1	3	2	0	0
3	1	2	3	1	0
2	0	0	1	3	0
5	1	1	0	3	0

Array as reordered by Algorithm B

Using a threshold value of 2 reveals two blocks or clusters :



Rows 4, 1 and 3 are related by virtue of the attributes corresponding to columns 1 and 4; and items (rows) 2 and 5 are similarly related by virtue of attribute (column) 3.

Fig. 1.3 - Example of Algorithm B

1.3 DISTANCE AND DISSIMILARITY

The previous algorithm has introduced the notion of distance - rows and columns were rearranged to be as "close" as possible. The great majority of clustering procedures are based on an initial definition of "closeness". The Euclidean (or "as-the-crow-flies") distance is widely used. If

$$\underline{a}_i = \{a_{ij} \mid j = 1,2,\ldots,m\} \quad , \text{ and}$$

$$\underline{a}_k = \{a_{kj} \mid j = 1,2,\ldots,m\}$$

are the i-th and k-th row-vectors of the data matrix, then the unweighted, squared Euclidean distance is given as :

$$d^2_{ik} = \sum_j (a_{ij}-a_{kj})^2 = (\underline{a}_i-\underline{a}_k)(\underline{a}_i-\underline{a}_k)^t = \|a_i\|^2 + \|a_k\|^2 - 2\underline{a}_i^t \cdot \underline{a}_k$$

where t denotes transpose, and $\|\cdot\|$ the norm with which the distance is associated. For n vectors, only n(n-1)/2 pairwise distances need to be considered in view of the symmetry ($d^2_{ik} = d^2_{ki}$).

Weighted versions of this distance are often used in order to remove the effects of very discrepant coordinate values (e.g. the case where some column j has values which are uniformly greater than other values). A standardized distance is defined by using centred and reduced values $(a_{ij}-\bar{a}_j)/\sigma_j$ instead of a_{ij}: $\bar{a}_j$ is the mean coordinate value over all the row-vectors,

$$\bar{a}_j = \sum_i a_{ij}/n$$

and σ_j is the standard deviation of the coordinate values:

$$\sigma^2_j = \sum_i (a_{ij} - \bar{a}_j)^2/n$$

Another distance is the Hamming (or "least-moves") distance, defined as:

$$d_{ik} = \sum_j |a_{ij} - a_{kj}|$$

and if the data array is binary, it is easily seen that the Hamming distance and the usual Euclidean distance squared yield identical results. The Hamming distance may be interpreted as the least set of "moves" required to reconfigure a binary string as another. It may also be easily generalized for qualitative or non-numeric data.

Given a set of characteristics, $j \in \{1,2,\ldots,m\}$, $|a_{ij} - a_{kj}| = 0$ if characteristic j for object i is identical to this characteristic for object k; if, on the other hand, the characteristics differ for these objects then $|a_{ij} - a_{kj}| = 1$. As an example, consider a database application where employee records have as attributes : department code, salary, age, length of service, etc. Next consider salaries to be categorized as 0-499, 500-1499, etc., and ages similarly categorized in five-years groups. Given two records (A, 1500, 33, 4, ...) and (A, 1300, 25, 2, ...), where the attribute values correspond to the attributes listed above, we have the contribution to the Hamming distance by these attributes as : 0+1+1+1+... .

Distances satisfy the following properties :

$d_{ik} = d_{ki}$ (symmetry)

$d_{ik} \geqslant 0$ if $i \neq k$
$d_{ik} = 0$ if $i = k$ } (positive semi-definiteness

$d_{ik} \leqslant d_{i\ell} + d_{k\ell}$ (triangular inequality).

The triangular inequality is easily verified in the Euclidean plane by considering the three vertices, i, k and ℓ, of a triangle. If the triangular inequality is not verified, we have a dissimilarity. Similarities (or proximities) are related to dissimilarities by subtraction or division. The Jaccard similarity, used with binary data, is defined as

$$s_{ik} = \frac{\#_j(a_{ij}=a_{kj}=1)}{\#_j(a_{ij}=1) + \#_j(a_{kj}=1) - \#_j(a_{ij}=a_{kj}=1)}$$

where # is the counting operator. Since $0 \leqslant s_{ik} \leqslant 1$, a dissimilarity may be defined by

$$d_{ik} = 1 - s_{ik}.$$

As an example, the Jaccard similarity of the vectors (10001001111) and (10101010111) is 5/(6+7-5).

A very large number of (dis)similarity coefficients have at one time or another been proposed. We will not attempt an inventory here : Anderberg (1973) remains a very readable introduction and Everitt (1980) may also be consulted.

1.4 GRAPH-THEORETIC APPROACHES

Without loss of generality, we will only consider the rows of the given data matrix since transposition will allow similar treatment of columns. Consider as given a dissimilarity, d, between all pairs of these objects. Three closely related analyses, using these dissimilarities, are the determining of connected components, the minimal spanning tree (MST), and the single linkage hierarchical clustering.

Connected components of the objects or data items (vertices) provide a partitioning of the object set (Fig. 1.4).

Algorithm C. Maximal connected subgraphs

Input	Set of $n(n-1)/2$ pairwise dissimilarities; threshold t.
Step 1	For each pair of objects, i and k, if $d_{ik} \leqslant t$ then place k on the component list associated with i.
Step 2	Take each non-empty component list (of object i, say) in turn, and carry out Step 2a.
Step 2a	Take each element, k, in turn in the component list of object i. If k has itself a non-empty component list, then append each element of k's component list to the component list of i, if these elements are not already present in i's component list; and then delete the component list of k.

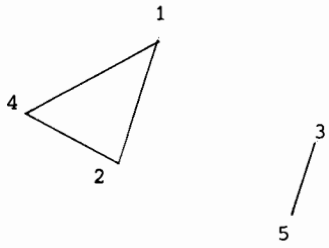
Step 1 sets up the graph of n vertices where adjacency between i and k is indicated by the presence of k in the linked list of i. Steps 2 and 2a constitute a breadth-first search for the maximal connected subgraphs, given a (possibly overlapping) set of connected subgraphs.

The minimal spanning tree may be related to Algorithm C : we seek the least threshold which allows each object (vertex) to be connected to some other vertex, with the constraint that no cycle arise in the graph.

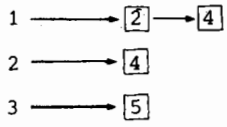
Some aspect of the MST is covered in most texts on clustering and on graph theory, and other areas besides. We will restrict ourselves to a brief description of this important method. The following algorithm constructs an MST by a "greedy" or nearest neighbour approach.

	1	2	3	4	5
1	0	4	9	5	8
2		0	6	3	6
3			0	6	3
4				0	5
5					0

Matrix of dissimilarities, d



Initial non-empty linked lists



Maximal connected subgraphs

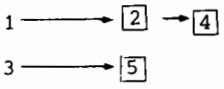


Fig. 1.4 - Connected components for threshold t = 5

Algorithm D. Minimal spanning tree

Step 1 Select an arbitrary vertex and connect it to the least dissimilar neighbour. These two vertices constitute a subgraph of the MST.

Step 2 Connect the current subgraph to the least dissimilar neighbour of any of the members of the subgraph.

Step 3 Loop on Step 2, until all vertices are in the one subgraph : this, then, is the MST.

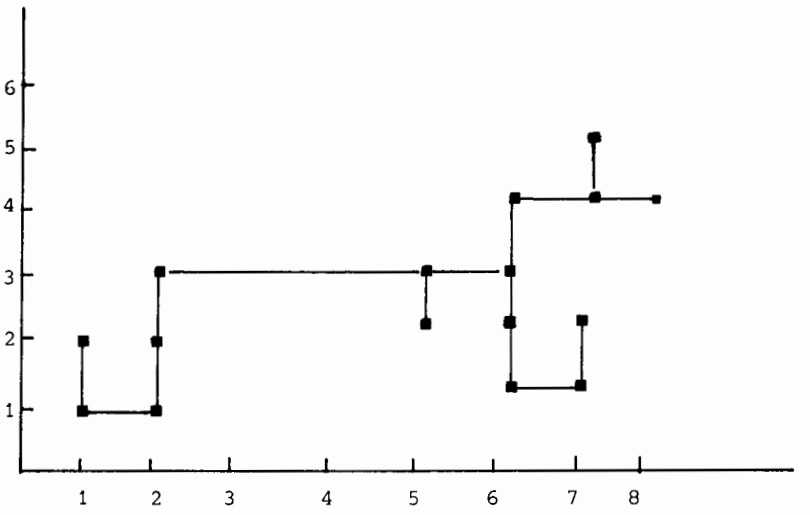
Step 2 agglomerates subsets of objects using the criterion of connectivity. For proof that Algorithm D does produce an MST, see for example Tucker (1980).

The following proposition should be apparent : breaking all links in the MST which are of length $\geq$ t yields components which are identical to the t-components obtained from Algorithm C. Thus, knowing the MST of an object-set allows sets of maximal connected subsets to be obtained for various thresholds. The reverse - constructing the MST from a set of connected subsets - is evidently not possible.

Obtaining components from an MST is the problem addressed by Zahn (1971). An edge is said to be inconsistent if it is of length much greater than the lengths of other nearby edges (see Fig. 1.5). Inconsistent edges may be picked out by looking at a histogram of edge lengths in the MST, and marking as deletable a set percentage of greatest-length edges. Alternatively, inconsistent edges may be obtained by defining a threshold of inconsistency such as being greater than two standard deviations above the mean length of all edges which are within two edges of incident vertices. Zahn applies these approaches to point pattern recognition - obtaining Gestalt patterns among sets of planar points; picking out bubble chamber particle tracks, where the curved and linear sequence of points are ideally suited to MST analysis; and finally detecting density gradients, where differing clusters of points have very different densities associated with them and hence are clearly distinguishable to the human eye. Some of these problems will be discussed in Chapter 4.

The use of the MST for storage reduction has also been proposed. Since the storage of university student records, or of chemical structures, will often involve a high degree of redundancy, it is suggested that the storage of indicators of where a new string of data differs from a previous one will reduce the overall storage required. Algorithm B may be used for this purpose. An alternative is to use the MST, where in addition we use a reference record - each record is related to some other, with the exception of a root record (see Fig. 1.6). A root in the MST, with the consequent defining of directed arcs from it, may be defined arbitrarily, but methods for choosing this privileged object - to be stored in its entirety - are discussed in Kang at al. (1977). A further extension which is discussed in the foregoing reference, and which might be more suitable in practice, is to use a spanning forest rather than the MST.

Finally, the MST is often suitable for outlier detection. Since outlying data items (resulting, perhaps, from typing errors in the inputting of attributes) will be of greater than average distance from their neighbours in the MST, they may be detected by drawing a histogram of edge lengths. Unusually large lengths will indicate the abnormal points (or data items) sought. Rohlf (1975) gives a statistical gap test, under the assumption that the edge lengths in the MST are normally distributed.



Edge connecting (2,3) and (5,3) is clearly inconsistent.

Fig. 1.5 - (Non-unique) minimal spanning tree of a point pattern

1.5 HIERARCHICAL AND NON-HIERARCHICAL METHODS

The single linkage hierarchical clustering approach outputs a set of partitionings of the object-set into maximal connected subgraphs, at each level - or for each threshold value which produces a new partition. It is therefore closely related to a sequence of outputs of Algorithm C, and it may also be obtained from an MST. The following algorithm, in its general structure, is relevant for a wide range of hierarchical clustering methods which vary only in the update formula used in Step 2. These methods may, for example, define a criterion of compactness in Step 2 to be used instead of the connectivity criterion used here.

Algorithm E. Single linkage hierarchical clustering

Input	An $n(n-1)/2$ set of dissimilarities.
Step 1	Determine the smallest dissimilarity, d_{ik}.
Step 2	Agglomerate objects i and k : i.e. replace them with a new object, iUk; update dissimilarities such that, for all objects $j \neq i,k$: $d_{iUk,j} = \min\{d_{ij}, d_{kj}\}$ Delete dissimilarities d_{ij} and d_{kj}, for all j, as these are no longer used.
Step 3	While at least 2 objects remain, return to Step 1.

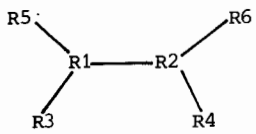
Record identifiers	Record contents
R1	a b a b a a a c a a
R2	a b a b a a a b a a
R3	b c a b a a a c a b
R4	a a a b a a a b a c
R5	a b a b a a a a a b
R6	a b a b a a a b a a

Store R2 as 1 8 b

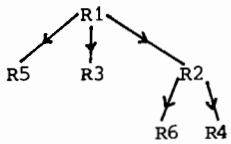
"As R1, except in location 8:b"
(saving 7 storage locations)

Store R3 as 1 1 b 2 c 10 b
(saving 3 locations)

Could R2 be reference point of R3 instead of R1 ?
Hamming distance between R1 and R3 : 3 ; between R2 and R3 : 4.
Hence it is better to have R1 as reference point.



Let R1 be root :



Hence, define R5, R3, R2 in terms of R1;
and R6 and R4 in terms of R2.

Fig. 1.6 - The use of the minimal spanning tree for storage compression

There are precisely n-1 agglomerations in Step 2 (allowing for a possibly arbitrary choice in Step 1 if there are a number of identical dissimilarities). It may be convenient to index the clusters found in Step 2 by n+1, n+2, ..., 2n-1, or an alternative practice is to index cluster iUk by the lesser of the indices of i and k. This latter convention is more appropriate if storage space is restricted: the replacement $d_{ij} \leftarrow \min\{d_{ij}, d_{kj}\}$ is made (where i is the lesser of indices i and k), so that the storage requirement never exceeds the space required for the initial matrix (this is done in Fig.1.7).

	1	2	3	4	5
1	0	4	9	5	8
2	4	0	6	3	6
3	9	6	0	6	3
4	5	3	6	0	5
5	8	6	3	5	0

Agglomerate 2 and 4 at dissimilarity 3

	1	2U4	3	5
1	0	4	9	8
2U4	4	0	6	5
3	9	6	0	3
5	8	5	3	0

Agglomerate 3 and 5 at dissimilarity 3

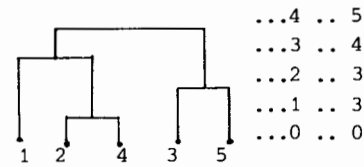
	1	2U4	3U5
1	0	4	8
2U4	4	0	5
3U5	8	5	0

Agglomerate 1 and 2U4 at dissimilarity 4

	1U2U4	3U5
1U2U4	0	5
3U5	5	0

Finally agglomerate 1U2U4 and 3U5 at dissimilarity 5

Resulting dendrogram



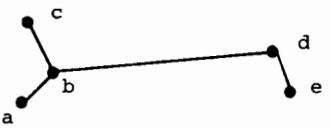
Ranks or levels

Criterion values (or linkage weights)

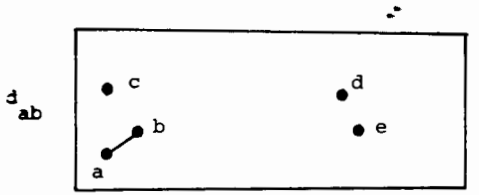
Fig. 1.7 - Single linkage dendrogram construction

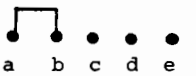
The title "single linkage" arises since, in Step 2, the interconnecting dissimilarity between two clusters (iUk and j) or components is defined as the least interconnecting dissimilarity between a member of one and a member of the other. Other hierarchical clustering methods are characterized by other functions of the interconnecting linkage dissimilarities.

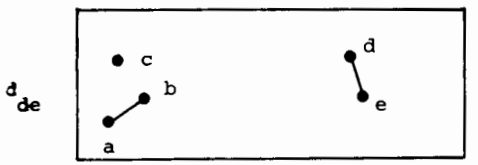
Since there are n-1 agglomerations, and hence iterations, and since Step 2 requires $< n$ operations, Algorithm E is of time complexity $O(n^2)$.

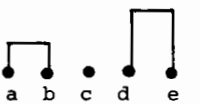


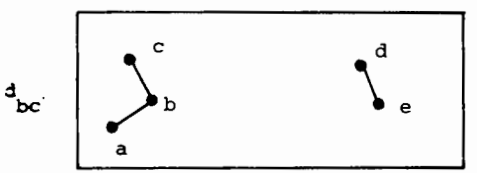
Minimal spanning tree

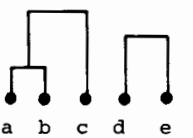


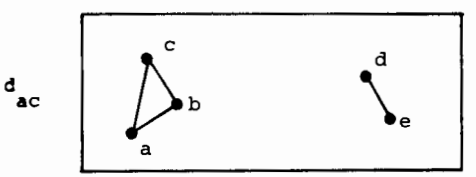


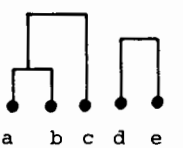


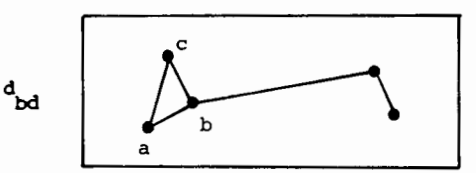


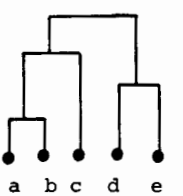












Ranked dissimilarities

Thresholded graph

dendrogram construction

Fig. 1.8 - Another approach to constructing a single linkage dendrogram

It is left as an exercise to devise an algorithm which takes an MST as input and produces a single linkage dendrogram. Although the single linkage hierarchy is an alternative representation of an MST, the reverse is not the case : in going from the MST to the hierarchy, the objects grouped together are noted but not the interconnecting edges which brought the grouping about.

Compared to the other hierarchic clustering techniques, the single linkage method can give rise to a notable disadvantage for summarizing interrelationships. This is known as chaining. An example is to consider four subject-areas, which it will be supposed are characterized by certain attributes : computer science, statistics, probability, measure theory. It is quite conceivable that "computer science" is connected to "statistics" at some threshold value, "statistics" to "probability" and "probability" to "measure theory", thereby giving rise to the fact that "computer science" and "measure theory" find themselves in the same cluster. This is due to the intermediaries "statistics" and "probability".

Rather than considering the very wide-ranging uses to which the single linkage method has been put, we will instead look at the general role played by a dendrogram, constructed by any **criterion.**

About 3/4 of all published work on clustering has employed hierarchical algorithms (Blashfield and Aldenderfer, 1978) : this figure might or might not hold for clustering usage, but it is nonetheless revealing. Interpretation of the information contained in a dendrogram will often be of one or more of the following kinds :

- set inclusion relationships,
- partition of the objects-set, and
- significant clusters.

We will briefly examine what each of these entail.

Much early work on hierarchic clustering was in the field of biological taxonomy, from the 1950s and more so from the 1960s onwards. The central reference in this area is Sokal and Sneath (1973). One major interpretation of hierarchies has been the evolution relationships between the organisms under study. It is hoped, in this context, that a dendrogram provides a sufficiently accurate model of underlying evolutionary progression. As an example, consider the hierarchical classification of languages based on certain features. It would be hoped to characterize clusters at higher levels of the tree as being language families, from which modern languages derive (Teutonic → German, English, Swedish, etc.). In another example of where dominance of inclusion relations are of importance, Johnson (1967) clusters

16 consonant sounds, where the dissimilarity is chosen to reflect confusability. Major groupings (voiced and unvoiced consonant phonemes, nasals) are found.

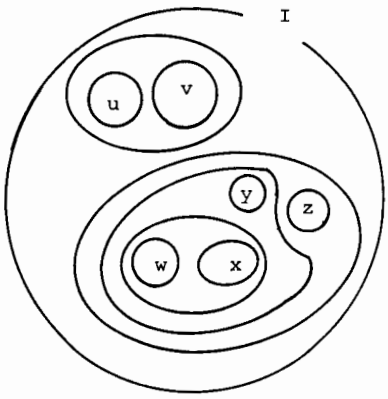
The most common interpretation made of hierarchic clustering is to derive a partition : a line is drawn horizontally through the hierarchy, to yield a set of classes. These clusters are precisely the connected components in the case of the single linkage method. A line drawn just above rank 3 (or criterion value 4) on the dendrogram in Fig.1.7. yields classes $\{1,2,4\}$ and $\{3,5\}$. Generally the choice of where "to draw the line" is arrived at on the basis of large changes in the criterion value. However, the criterion value is usually increasing towards the final set of agglomerations, which renders the choice of best partition on this basis difficult. Since every line drawn through the dendrogram defines a partition, it is often expedient to choose a partition with convenient features (number of classes, numbers of objects per class). Various automatic procedures have been proposed for cutting a dendrogram, and a bibliography and appraisal of many of these is given by Milligan and Cooper (1983).

A final type of interpretation, less common than the foregoing, is to dispense with the requirement that the classes chosen constitute a partition, and instead detect maximal (i.e. disjoint) clusters of interest at varying levels of the hierarchy. Such an approach is used by Rapoport and Fillenbaum (1972) in a clustering of colours based on semantic attributes.

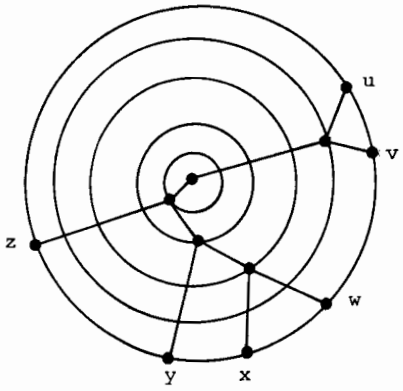
In summary, a dendrogram provides a résumé of many of the proximity and classificatory relationships in a body of data. For an experienced user, it is a convenient representation which answers such questions as : "How many groups are in this data ?", "What is the most suitable grouping of these objects ?", "What are the salient interrelationships present ?". But for an inexperienced user, it should be stressed that differing answers can be feasibly provided by a dendrogram for most of these questions, just as different human observers would also arrive at different outcomes.

We will conclude this section with a short look at non-hierarchical clustering methods.

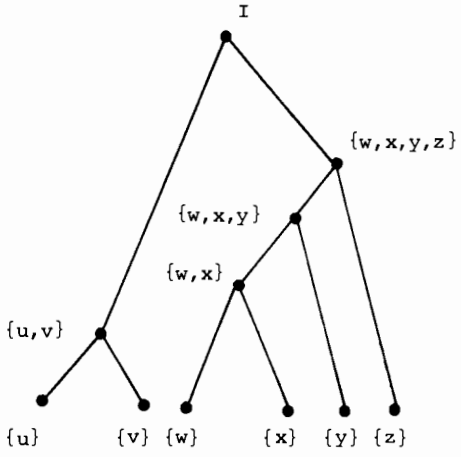
A large number of assignment algorithms have been proposed. The single-pass approach usually achieves computational efficiency at the expense of precision, and there are many iterative approaches for improving on crudely-derived partitions.



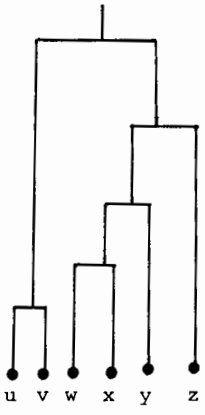
a) Embedded sets



b) "Sky-view" (a particular representation of the minimal spanning tree)



c) Tree



d) Dendrogram

```
u   v   w   x   y   z
.   .   .   .   .   .
x x x
x x x   x x x
x x x   x x x x x
x x x   x x x x x x x
x x x x x x x x x x x
```

e) Sky-line plot

Fig. 1.9 - Differing representations of a hierarchic clustering on object set I = {u,v,w,x,y,z}

As an example of a single-pass algorithm, Algorithm F is given in Salton and McGill (1983). The general principle followed is : make one pass through the data assigning each object to the first cluster which is close enough and making a new cluster for objects that are not close to any existing cluster.

Algorithm F	Single-pass overlapping cluster algorithm
Input	n objects, threshold t, dissimilarity on objects.
Step 1	Read object 1, and insert object 1 in membership list of cluster 1. Let representative of cluster 1 be given by object 1. Set i to 2.
Step 2	Read i-th object. If d(i-th object, cluster j) $\leq$ t, for any cluster j, then include the i-th object in the membership list of cluster j, and update cluster representative vector to take account of this new member. If d(i-th object, cluster j) $>$ t, for all clusters j, then create new cluster, placing i-th object in its membership list, and letting representative of this cluster be given by the i-th object.
Step 3	Set i to i+1. If i $<$ n, go to Step 2.

The cluster representative vector used is usually the mean vector of the cluster's members; in the case of binary data, this representative might then be thresholded to have 0 to 1 coordinate values only. In Step 2, it is clear that overlapping clusters are possible in the above algorithm. In the worst case, if threshold t is chosen too low, all n objects will constitute clusters and the number of comparisons to be carried out will be $O(n^2)$. The dependence of Algorithm F on the given sequence of objects is the price paid for performance which will generally be much faster than this worst case order-of-magnitude performance.

Many algorithms have been proposed which permit certain object-vectors to be defined as initial cluster representatives, or choose such cluster seeds at random. The cluster centres are then recalculated in the light of the assignments.

The rereading of all cases, and reassigning to the new cluster representatives, may be iterated until the cluster representatives show sufficiently small alteration in successive passes. Such iterative readjustment is a common feature of many partitioning (or non-hierarchical) clustering algorithms. Either relocation or exchange of cluster

members may also follow the initial assignment to clusters. An overall criterion function determines whether a proposed relocation (or exchange) will be carried out. A difficulty with iterative algorithms is the requirement for parameters to be set in advance (Anderberg, 1973, describes a version of ISODATA requiring 7 pre-set parameters). As a broad generalization, it may thus be asserted that iterative algorithms ought to be considered when the problem is clearly defined in terms of numbers and other characteristics of clusters; but that hierarchical routines offer a more general-purpose and user-friendly option.

1.6 REFERENCES

The one-dimensional ordering of rows and columns of an array is used by : S.B. Deutsch and J.J. Martin, An ordering algorithm for analysis of data arrays, Operations Research, 19, 1350-1362 (1971)

O. Egan, Affective development in adolescent conceptions of Ireland, The Irish Journal of Education 11, 61-73 (1977)

The clustering by reordering of rows and columns is dealt with by : W.T. Cormick, P.J. Schweitzer and T.J. White, Problem decomposition and data reorganization by a clustering technique, Operations Research 20, 993-1009 (1972)

S.T. March, Techniques for structuring database records, Computing Surveys 15, 45-79 (1983)

Distances and dissimilarities are examined in :
M.R. Anderberg, Cluster Analysis for Applications, Academic Press, New York (1973)

B. Everitt, Cluster Analysis, Heineman Educational Books, London, (1980) 2nd edition

For minimal spanning trees, see :
A. Tucker, Applied Combinatorics, Wiley, New York (1980)

C.T. Zahn, Graph-theoretical methods for detecting and describing Gestalt clusters, IEEE Transactions on Computers C-20, 68-86 (1971)

A.N.C. Kang, R.C.T. Lee, C.L. Chang and S.K. Chang, Storage reduction through minimal spanning trees and spanning forests, IEEE Transactions on Computers C-26, 425-434 (1977)

F.J. Rohlf, Generalization of the gap test for the detection of multivariate outliers, Biometrics 31, 93-101 (1975)

The first of the following references is a seminal work for biological applications :

P.H.A. Sneath and R.R. Sokal, Numerical Taxonomy, Freeman, San Francisco (1973)

S.C. Johnson, Hierarchical clustering schemes, Psychometrika 32 , 241-254 (1967)

G.W. Milligan and M.C. Cooper, An examination of procedures for determining the number of clusters in a data set, Working Series Papers 83-51, College of Administrative Science, The Ohio State University, 28 pp. (1983)

A. Rapoport and S. Fillenbaum, An experimental study of semantic structures, in: A.K. Romney, R.N. Shepard and S.B. Nerlove (Eds.), Multidimensional Scaling : Theory and Applications in the Behavioural Sciences, Seminar Press, New York, pp. 93-131 (1972)

G. Salton and M.J. McGill, Introduction to Modern Information Retrieval, McGraw-Hill, New York (1983)

Three broad-ranging books which will repay close study are :

J.A. Hartigan, Clustering Algorithms, Wiley, New York (1975)

J.P. Benzécri et al., L'analyse des Données. I. Taxinomie, Dunod, Paris (1979) 3rd Edition

H.H. Bock, Automatische Klassifikation, Vandenhoeck und Rupprecht, Göttingen (1974)