COMPSTAT LECTURES 4

lectures in computational statistics

physica-verlag · würzburg-wien

ISSN 0172–6528

COMPSTAT LECTURES 4

lectures in computational statistics

Edited by
J. M. Chambers, J. Gordesch, A. Klas, L. Lebart, and P. P. Sint

Physica-Verlag · Vienna—Würzburg 1985

ISSN 0172—6528 ISBN 3 7051 0008 4

Printed by repro-druck Liebing GmbH, Würzburg

ISBN 3 7051 0008 4

Multidimensional Clustering Algorithms

Fionn Murtagh

Space Telescope – European Coordinating Facility

To my mother and father.

P R E F A C E

The objectives of this monograph are as follows: to collect together important recent algorithmic results in the area of cluster analysis; to indicate algorithms which may be of importance in parallel computing environments; to include (unlike other general texts on clustering) discussion of problems specific to the computing area such as pattern recognition and information storage and retrieval; and to clearly describe clustering algorithms which are of general, practical relevance.

Chapter 1 provides an introduction to this area. Little background knowledge is presupposed, with the exception of some familiarity with terminology and problems in the fields of graph theory and of algorithmics.

This monograph was written during my stay in 1984 as Visiting Scientist in the Information Analysis and Data Handling Division of the Joint Research Centre of the Commission of the European Communities at Ispra, Italy. My thanks go to Dr. Larisse for organizing this stay and to Dr. Colombo for much support. My gratitude also goes to Signora Giaretta whose efficient typing service considerably eased the burden of preparing this text.

Fionn Murtagh, Ispra, October 1984

COMPSTAT LECTURES

Lectures in Computational Statistics

Edited by J.M. Chambers, J. Gordesch, A. Klas, L. Lebart, and P.P. Sint

ISSN 0172-6528

The main purpose of COMPSTAT - LECTURES is to publish authoritative papers and surveys relevant to areas of computational statistics which are in vigorous development. Preferably, papers should be of broad interest, and offer unified presentation of some subject not readily available in the existing statistical literature.

Vol. 1
P. Naeve, CAI and Computational Statistic. – **G. Pflug**, Some Remarks on Stochastic Approximation. – **H. Skarabis**, Introduction to Multidimensional Scaling. – **H. Skarabis**, Multidimensional Scaling with prescribed Structures. – **P.P. Sint**, Cluster Analysis, an Introduction. – **H. Skarabis et al.**, Sequentializing Nonparametric Tests. – **J. Gordesch**, System Simulation. – **J. Gordesch**, Programming Abstract Structures in Statistics and Probability.
1978. 132 pp. Softcover DM 30.–
ISBN 3 7908 0197 6.

Vol. 2
M. Ribarič et al., Computational Methods for Parsimonious Data Fitting.
1984. 154 pp. Softcover DM 58.–
ISBN 3 7051 0004 1.

Vol. 3
A. Berlinet, Estimating the Degrees of an Arma Model.
P. Ihm and **H. van Groenewoud**, Correspondence Analysis and Gaussian Ordination.
1984. 94 pp. Softcover DM 44.–
ISBN 3 7051 0006 8.

Physica-Verlag · Wuerzburg–Vienna

Place your order directly with:

Physica-Verlag · P.O.Box 5840
D–8700 Wuerzburg/Germany

ISBN 3 7051 0008 4

CONTENTS

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.

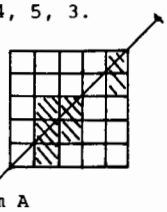
3	0	0	2	1
0	0	3	1	0
2	0	1	3	1
3	1	0	1	0
1	0	3	0	1

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.

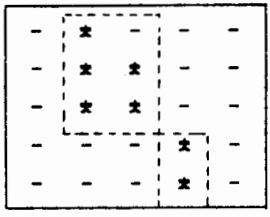
	1	2	3	4	5
1	3	0	0	2	1
2	0	0	3	1	0
3	2	0	1	3	1
4	3	1	0	1	0
5	1	0	3	0	1

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.

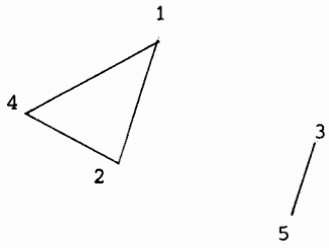
Step 1 sets up the graph of n vertices where adjacency between i and k is indicated by the presence of k in the linked list of i. Steps 2 and 2a constitute a breadth-first search for the maximal connected subgraphs, given a (possibly overlapping) set of connected subgraphs.

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.

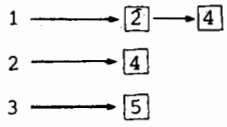
Some aspect of the MST is covered in most texts on clustering and on graph theory, and other areas besides. We will restrict ourselves to a brief description of this important method. The following algorithm constructs an MST by a "greedy" or nearest neighbour approach.

	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

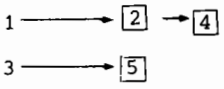


Fig. 1.4 - Connected components for threshold t = 5

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.

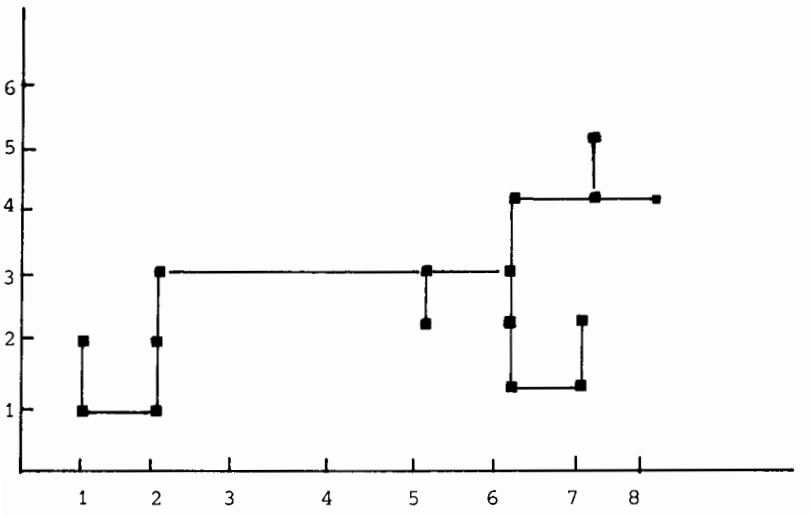
Step 2 agglomerates subsets of objects using the criterion of connectivity. For proof that Algorithm D does produce an MST, see for example Tucker (1980).

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.

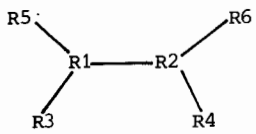
Record identifiers	Record contents
R1	a b a b a a a c a a
R2	a b a b a a a b a a
R3	b c a b a a a c a b
R4	a a a b a a a b a c
R5	a b a b a a a a a b
R6	a b a b a a a b a a

Store R2 as 1 8 b

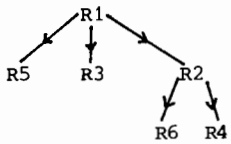
"As R1, except in location 8:b"
(saving 7 storage locations)

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

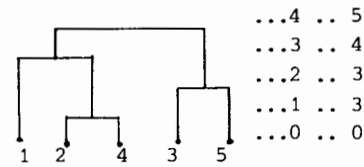
	1	2U4	3U5
1	0	4	8
2U4	4	0	5
3U5	8	5	0

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



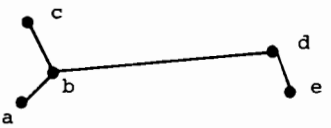
Ranks or levels

Criterion values (or linkage weights)

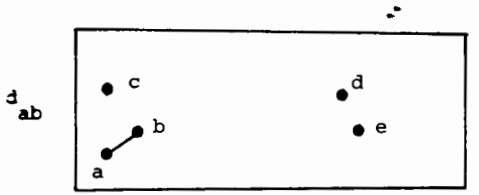
Fig. 1.7 - Single linkage dendrogram construction

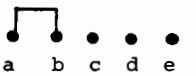
The title "single linkage" arises since, in Step 2, the interconnecting dissimilarity between two clusters (iUk and j) or components is defined as the least interconnecting dissimilarity between a member of one and a member of the other. Other hierarchical clustering methods are characterized by other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

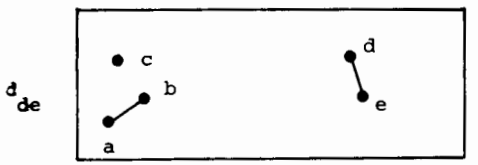
Since there are n-1 agglomerations, and hence iterations, and since Step 2 requires $< n$ operations, Algorithm E is of time complexity $O(n^2)$.

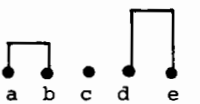


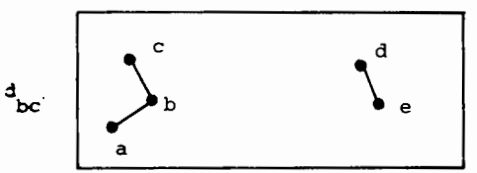
Minimal spanning tree

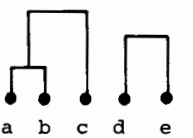


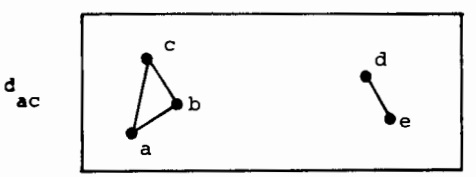


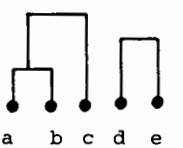


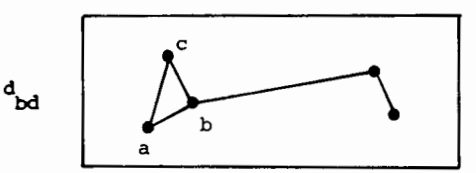


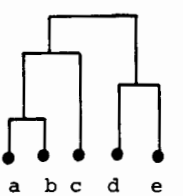












Ranked dissimilarities

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.

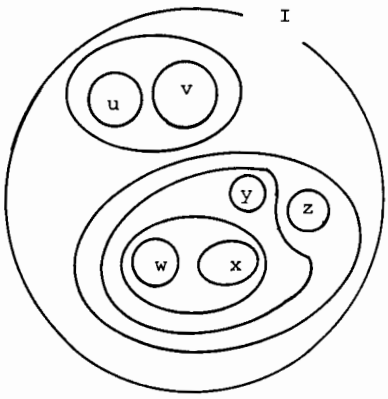
The most common interpretation made of hierarchic clustering is to derive a partition : a line is drawn horizontally through the hierarchy, to yield a set of classes. These clusters are precisely the connected components in the case of the single linkage method. A line drawn just above rank 3 (or criterion 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.

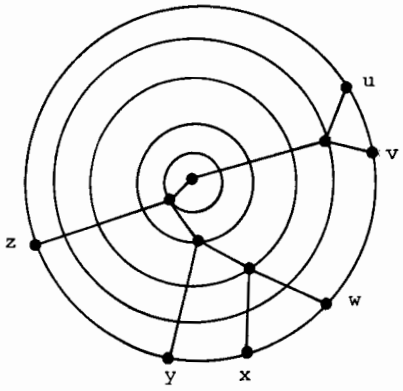
In summary, a dendrogram provides a résumé of many of the proximity and classificatory relationships in a body of data. For an experienced user, it is a convenient representation which answers such questions as : "How many groups are in this data ?", "What is the most suitable grouping of these objects ?", "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.

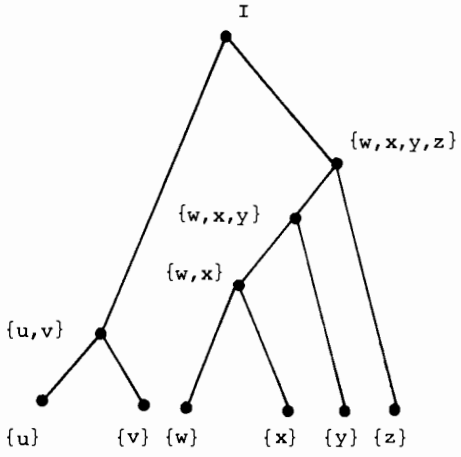
A large number of assignment algorithms have been proposed. The single-pass approach usually achieves computational efficiency at the expense of precision, and there are many iterative approaches for improving on crudely-derived partitions.



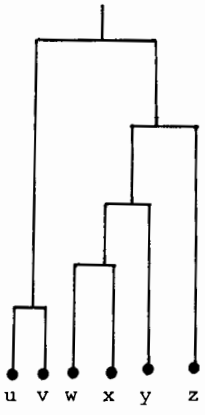
a) Embedded sets



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)

CHAPTER 2 - FAST NEAREST NEIGHBOUR SEARCHING

2.1 INTRODUCTION

Given a set of n multidimensional vectors, the nearest neighbour (NN) problem is to find the closest vector to some given vector. Efficient NN-finding algorithms may be incorporated into clustering algorithms, as will be seen in Chapters 3 and 4. NN-finding algorithms can be made the basic "building blocks" of many commonly-used clustering algorithms, and thus progress in the area of NN-finding has immediate beneficial repercussions on clustering.

The NN problem is also important in its own right in other areas. In discriminant analysis in pattern recognition, a k-nearest neighbour assignment strategy is widely used : an unlabelled sample is assigned to the class to which a majority of its NNs belong (see Kittler, 1978). In database design, the NN problem is known as the best match problem : the closest record to a query is to be accessed efficiently (see Bentley, 1979). In information retrieval, a similar problem arises when the NN document of a query or request vector is required (see Perry and Willett, 1983). In such problems as minimizing head movement on direct access I/O devices, or the optimal sequencing of pens for plotting devices, a NN-based algorithm has been found to be both simple and effective (see Bentley, 1984).

Sections 2.2 and 2.3 both describe particular data structures where the objective is to break the O(n) barrier for determining the NN of a point, - in the average case if not in the worst case. These approaches have been very succeessful, but they are restricted to low dimensional NN-searching. For higher-dimensional data, a wide range of bounding approaches have been proposed, i.e. a theoretically smallest possible dissimilarity (the lower bound on the dissimilarity) is compared against the current NN dissimilarity, and if the former is the greater then there is no need to proceed to an exact calculation of the dissimilarity. Bounding approaches

remain O(n) algorithms, but with a low constant of proportionality such algorithms can be very efficient.

Bounding approaches of a general nature are discussed in sections 2.4 and 2.5. The general principle of the branch and bound algorithm described in section 2.5 appears to be particularly versatile for general purpose applications. In section 2.6, the particular problem-area of information retrieval is focussed on. Finally, section 2.7 indicates what practical problems require attention at the present time.

2.2 Hashing

In this approach to NN-searching, a preprocessing stage precedes the searching stage. All points are mapped onto indexed cellular regions of space, so that NNs will be found in the same or in closely adjacent cells. Taking the plane as an example, and considering points (x_1, y_1), the maximum and minimum values on all coordinates are obtained (e.g. x_{min}, x_{max}). The mapping $x_i \rightarrow \lfloor (x_i - x_{min})/r \rfloor$, where constant r is chosen in terms of the number of equally spaced categories into which the interval $[x_{min}, x_{max})$ is to be divided, gives an integer value between 0 and $\lfloor (x_{max} - x_{min})/r \rfloor$ to x_i. Defining a similar mapping for y_i allows each point to be transformed onto a rectangular cell (see Fig. 2.1). O(nm) time is required to obtain the transformation of all n points in $\mathbb{R}^m$. The result is stored as a linked list with a pointer from each cell identifier to the set of all points which have been mapped onto that cell.

The implementation of NN searching proceeds as follows. Firstly, the closest point which is mapped onto the same grid cell as the target point is found. This will be a current NN point. A closer point may be mapped onto some other grid cell if the distance between target point and current NN point is greater than the distance between the target point and any of the boundaries of the cell containing it. If this is the case, all grid cells adjacent to the grid cell are searched in order to see if the current NN point can be bettered. It is, of course, possible that the initial cell might have contained no points other than the target point: in this case, the search through adjacent cells may yield a possible NN, and it may be necessary to widen the search to confirm this. Therefore in $\mathbb{R}^2$, 1 cell is searched first; if required, the 8 cells adjacent to this are searched; if necessary, the 16 cells adjacent to these cells (or less if restrained by the exterior walls of the region in which the search is taking place) are then searched; and so on.

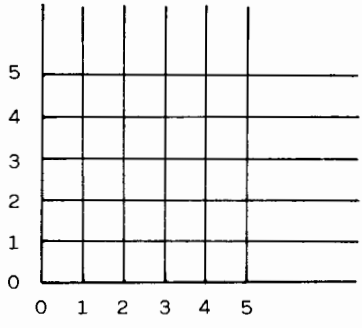
A powerful theoretical result regarding this approach is as follows: for uniformly distributed points the NN of a point is found in O(1) expected time (see Delannoy, 1980, or Bentley et al., 1980, for proof). Therefore this approach will work well if approximate uniformity can be assumed or if the data can be broken down into regions of approximately uniformly distributed points.

The search through adjacent cells requires time which increases exponentially with dimensionality (if it is assumed that the number of points assigned to each cell is approximately equal). As a result, this approach is suitable for low dimensions only. Rohlf (1978) reports on work in dimensions 2, 3 and 4, and Murtagh (1983) in $\mathbb{R}^2$. Rohlf also mentions the use of the first 3 principal components to approximate a set of points in 15-dimensional space.

x_{min} , y_{min} = 0

x_{max} , y_{max} $\leqslant$ 50

r = 10



Point (21,40) is mapped onto cell (2,4)

Point (7,11) is mapped onto cell (0,1)

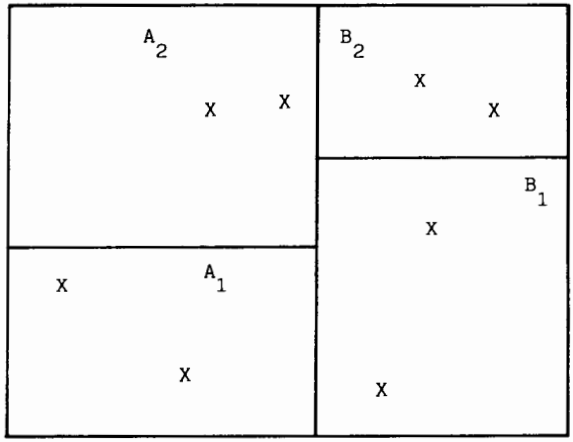
Fig. 2.1 - Example of hashing in $\mathbb{R}^2$.

2.3 MULTIDIMENSIONAL BINARY SEARCH TREE

A decision tree splits the data to be searched through into 2 parts; each part is further subdivided; and subdivisions continue until some prespecified number of data points is arrived at. In practice, we associate with each node of the decision tree the definition of a subdivision of the data only, and we associate with each terminal node a pointer to the stored coordinates of the points (perhaps on direct access storage).

One version of the multidimensional binary search tree (MDBST) is as follows. Halve the set of points, using the median of the first coordinate values of the points. For each of the resulting sets of points, halve them using the median of the second coordinate values. Continue halving; use the medians of all coordinates in succession in order to define successive levels of the hierarchical decomposition; when all coordinates have been exhausted, recycle through the set of coordinates; halt when the number of points associated with the nodes at some level is smaller than or equal to a prespecified constant, c. See example in Fig. 2.2.
Clearly the tree is kept balanced : therefore there are O(log n) levels, at each of which O(n) processing is required. Hence the construction of the tree takes O(n log n) time.

The search for a NN then proceeds by a top-down traversal of the tree : the target point is transmitted through successive levels of the tree using the defined separation of the two child nodes at each node. On arrival at a terminal node, all associated points are examined and a current NN selected. The tree is then backtracked : if the points associated with any node could furnish a closer point, then subnodes must be checked out.



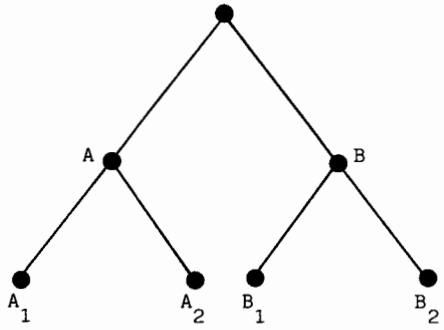


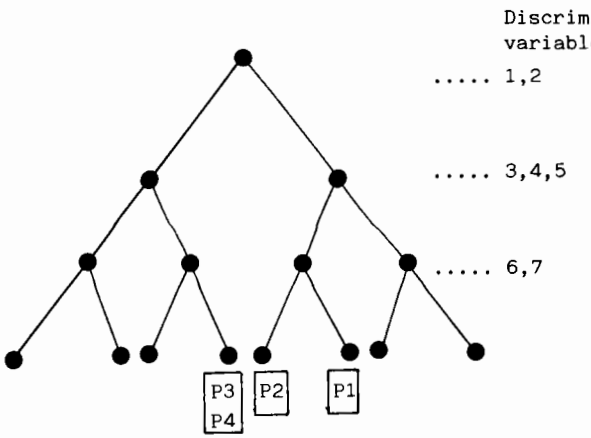
Fig. 2.2 - MDBST in $\mathbb{R}^2$.

The approximately constant number of points associated with terminal nodes (i.e. with hyper-rectangular cells in the space of points) should be greater than 1 in order that some NNs may be obtained without requiring a search of adjacent cells (i.e. of other terminal nodes). Friedman et al. (1977) suggest a value of c between 4 and 32 based on empirical study.

The MDBST approach only works well with small dimensions. To see this, consider each coordinate being used once only for the subdivision of points (i.e. each variable is considered equally useful). Let there be p levels in the tree, i.e. 2^p terminal nodes. Each terminal node contains approximately c points by construction and so $c.2^p = n$. Therefore $p = \log_2 n/c$. As sample values, if n = 32768, c = 32, then p = 10; i.e. in 10-dimensional space, using a large number of points associated with terminal nodes, more than 30000 points will need to be considered. For higher dimensional spaces, two alternative MDBST specifications are as follows.

All variables need not be considered for splitting the data if it is known that some are of greater interest than others. Linearity present in the data may manifest itself via the variance of the variables; choosing the variable with greatest variance as the discriminator variable at each node may therefore allow repeated use of certain variables. This has the added effect that the hyperrectangular cells into which the terminal nodes divide the space will be approximately cubical in shape. In this case, Friedman et al. (1977) show that search time is O(log n) on average for the finding of the NNs of all n points. Results obtained for dimensionalities of between 2 and 8 are reported on in Friedman et al. (1977), and in the application of this approach to minimal spanning tree (cf. Chapter 4) construction in Bentley and Friedman (1978).

In information retrieval, dimensionality is often greater than the number of points. Point coordinates here indicate the association or non-association (1 or 0) of an index term (variable) with a document (point). In this context, variables may be batched together. In Fig. 2.3 a target point with 0 coordinate values on attributes 1 and 2 is directed towards the left child node of the root; otherwise it is sent to the right child node; at level 2 if the target point has 0 coordinates for attributes 3, 4 and 5, it is directed towards the left subtree and otherwise towards the right subtree. As with the version of the MDBST described above, a top-down traversal of the tree leads to the search of all points associated with a terminal node; this provides a current NN; then backtracking is commenced in order to see if the current NN can be bettered. The maximum size data set for which this approach has been used is 1400 documents. Results obtained have not been good (90% of documents required searching) but it is suggested that greater n would be more successful (Weiss, 1981; Eastman and Weiss, 1982). General guidelines for the variables which define the direction of search at each level are that they be related, and the number chosen should keep the tree balanced.



		Variables							
		1	2	3	4	5	6	7	8
Points	P1	1	0	0	0	0	1	1	1
	P2	1	1	0	0	0	0	0	0
	P3	0	0	1	0	0	1	1	0
	P4	0	0	0	1	1	1	0	0

Decision rule : presence of some one of discriminator variables

⇒ take right subtree.

Fig. 2.3 - MDBST using batching of variables for binary data.

2.4 BOUNDING USING PROJECTIONS OR THE EUCLIDEAN DISTANCE

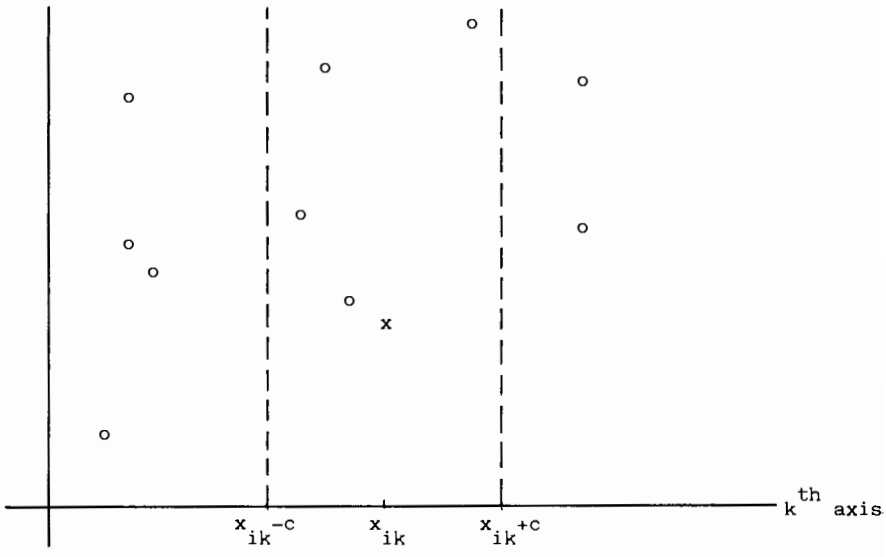
All remaining algorithms to be discussed make use of bounds : some lower bound on the dissimilarity is efficiently calculated in order to dispense with the full calculation in many instances.

Using projections on a coordinate axis allows the exclusion of points in the search for the NN of point x_i. Points x_j, only, are considered such that $(x_{ik} - x_{jk})^2 \leqslant c$, where x_{ik} (x_{jk}) is the k^{th} coordinate of x_i (x_j) and c is a prespecified constant (see Fig. 2.4).

Alternatively, more than one coordinate may be used. The prior sorting of coordinate values on the chosen axis or axes will expedite the finding of points whose full distance calculation is necessitated. The preprocessing required with this approach involves the sorting of up to m sets of coordinates, i.e. O(mn log n) time.

Using one axis, it is evident that many points may be excluded if the dimensionality is very small, but that the approach will disimprove as the latter grows. Friedman et al. (1975) give the expected NN search time, under the assumption that the points are uniformly distributed, as $O(m\, n^{1-1/n})$. This approaches the brute force O(nm) as m gets large. Reported empirical results are for dimensions 2 to 8.

Marimont and Shapiro (1979) extend this approach by the use of projections in subspaces of dimension greater than 1 (usually about m/2 is suggested). This can be further improved if the subspace of the principal components (following a principal component analysis) is used. Dimensions up to 40 are examined.



Points with projections within distance c of given point's ("x") projection, alone, are searched.

Fig. 2.4 - Two-dimensional example of projection-based bound.

The Euclidean distance is the (dis)similarity which is most widely used in practice. Two other Minkowski distances require less computation time to calculate and provide bounds on the Euclidean distance. We have :

$$d_1(x,x') \geqslant d_2(x,x') \geqslant d_\infty(x,x')$$

where d_1 is the Hamming or "city block" distance, defined as $\sum_j |x_j - x'_j|$; the Euclidean distance is given by the square root of $\sum_j (x_j - x'_j)^2$; and the Chebyshev distance is defined as $\max_j |x_j - x'_j|$.

Kittler (1978) makes use of the following bounding strategy: reject all points y such that $d_1(x,y) \geqslant \sqrt{m}.\delta$ where δ is the current NN d_2-distance, and m is the dimensionality of the space. The more efficiently calculated d_1-distance may thus allow the rejection of many points (90% in 10-dimensional space in reported by Kittler). Kittler's rule is obtained by noting that the greatest d_1-distance between x and x' is attained when

$$|x_j - x'_j|^2 = \frac{d_2^2(x,x')}{m}$$

for all coordinates j; hence $d_1(x,x') = d_2(x,x')/\sqrt{m}$ is the greatest d_1-distance between x and x'. In the case of the rejection of point y, we then have :

$$d_1(x,y) \leqslant d_2(x,y)/\sqrt{m}$$

and since

$$d_1(x,y) \geqslant \sqrt{m}\,\delta$$

by virtue of the rejection, it follows that

$$\delta \leqslant d_2(x,y)$$

Yunck (1976) presents a theoretical analysis for the similar use of the Chebyshev metric. Richetin et al. (1980) propose the use of both bounds. Using uniformly distributed points in dimensions 2 to 5, the latter reference reports the best outcome when the rule : reject all y such that $d_\infty(x,y) \geqslant \delta$ precedes the rule based on the d_1-distance. Up to 80% reduction in CPU time is reported.

2.5 BOUNDING USING THE TRIANGULAR INEQUALITY

The triangular inequality is satisfied by distances : $d(x,y) \leqslant d(x,z)+d(z,y)$ where x,y and z are any three points. The use of a reference point, z, allows a full distance calculation between point x (whose NN is sought) and y to be avoided if

$$|d(y,z)-d(x,z)| \geqslant \delta$$

where δ is the current NN distance. The set of all distances to the reference point are calculated and stored in a preprocessing step requiring O(n) time and O(n) space. The above cut-off rule is obtained by noting that if

$$d(x,y) \geqslant |d(y,z)-d(x,z)|$$

then, necessarily, $d(x,y) \geqslant \delta$. The former inequality above reduces to the triangular inequality irrespective of which of d(y,z) or d(x,z) is the greater.

The set of distances to the reference point, $\{ d(x,z) | x \}$, may be sorted in the preprocessing stage. Since d(x,z) is fixed during the search for the NN of x, it follows that the cut-off rule will not then need to be applied in all cases.

The single reference point approach, due to Burkhard and Keller (1973), was generalized to multiple reference points by Shapiro (1977). The sorted list of distances to the first reference point, $\{d(x,z_1)|x\}$, is used as described above as a preliminary bound. Then the subsequent bounds are similarly employed to further reduce the points requiring a full distance calculation. The number and the choice of reference points to be used is dependent on the distributional characteristics of the data. Shapiro (1977) finds that reference points ought to the located away from groups of points.

In 10-dimensional simulations, it was found that at best only 20% of full distance calculations were required (although this was very dependent on the choice of reference points).

Fukunaga and Narendra (1975) make use of both a hierarchical decomposition of the data set (they employ repeatedly the k-means iterative/relocatory algorithm), and bounds based on the triangular inequality. For each node in the decomposition tree, the centre and maximum distance to the centre (the "radius") of associated points are determined. For 1000 points, the above reference uses 3 levels where there was a division into 3 classes at each node.

All points associated with a non-terminal node can be rejected in the search for the NN of point x if the following rule (Rule 1) is not verified:

$$d(x,g) - r_g < \delta$$

where δ is the current NN distance, g is the centre of the cluster of points associated with the node, and r_g is the radius of this cluster. For a terminal node, which cannot be rejected on the basis of this rule, each associated point, y, can be tested for rejection using the following rule :

$$| d(x,g)-d(y,g) | \geqslant \delta.$$

These two rules are direct consequences of the triangular inequality. A branch and bound algorithm may be implemented using these 2 rules. This involves determining some current NN (the bound) and subsequently "branching" out of a traversal path whenever the current NN cannot be bettered. The control structure for a backtracking implementation of this is shown in Fig. 2.5. The current NN distance will be initialized to machine infinity, and subsequently updated.

This NN searching approach appears particularly promising for general-purpose applications. It is also not inherently limited by dimensionality considerations.

```
node ← root.
call FUKNAR (node).
-
-
-
FUKNAR (node) :
if node = terminal then call TERM (node).
                   else if Rule-1-verified then node ← left-subnode;
                                                call FUKNAR (node).
node ← next-subnode-from-the-left; call FUKNAR (node).
end
```

FUKNAR is a recursive procedure to implement backtracking NN search.

TERM is a procedure to carry out Rule 2 search of all points associated with the terminal node.

Rule-1-verified may be implemented by a procedure call.

The assignments to variable "node" require a choice of storage structure for the tree, and an associated set of flags to indicate when a node is not to be reused.

Fig. 2.5 - Control structure for backtracking NN searching.

2.6 NEAREST NEIGHBOUR ALGORITHMS IN INFORMATION RETRIEVAL

In information retrieval, m may typically be of the same order of magnitude as n or greater. Values in excess of 10,000 are not uncommon for test collections. The particular nature of the data (very sparse binary document-term associations) has given rise to particular NN-searching algorithms. The data is usually stored on direct access storage as compact linked lists (i.e. the sequence number of the document is followed by the sequence numbers of the terms which are associated with it). Current commercial document collections are usually searched using a Boolean search environment (i.e. all documents associated with particular terms are retrieved; the intersection/union/etc. of such sets of documents are obtained using AND/OR and other connectives). For efficiency, an inverted file which maps terms onto documents is available for Boolean retrieval, - stored in a similar manner to the document-term file. Although storage considerations are clearly of paramount importance, it might grosso modo be stated that we have available both the document-term matrix, and its transpose. The efficient NN-searching algorithms, to be discussed, will make use of both document-term and term-document files.

The usual algorithm for NN-searching considers each document in turn, calculates the dissimilarity with the given document (or query, which mathematically, is identical to a document), and updates the current NN if necessary. This algorithm is shown schematically in Fig. 2.6. The inner loop is simply an expression of the fact that the dissimilarity will (in general) require O(m) calculations: examples of commonly-used coefficients are the Jaccard similarity, i.e. $\#(i \cap i')/\#(i \cup i')$, and the Hamming distance, i.e. $\#(i) + \#(i') - 2\#(i \cap i')$. Here, # is the counting operator, indicating the number of terms associated with document i(i.e. $\#(i)$) or the number of terms shared by documents i and i' (i.e. $\#(i \cap i')$).

Usual algorithm -

```
Initialize current NN.
For all documents in turn do :
... For all terms associated with the document do :
... ..... Determine dissimilarity
... End.
... Test against current NN.
End.
```

Croft's algorithm -

```
Initialize current NN.
For all terms associated with the given document do :
... For all documents associated with each term do :
... ..... For all terms associated with a document do :
... ..... ..... Determine dissimilarity.
... ..... End.
... ..... Test against current NN.
... End.
End.
```

Perry-Willett algorithm -

```
Initialize current NN.
For all terms associated with the given document (i) do :
... For all documents (i') associated with each term do :
... ..... Increment location i' of counter vector.
... End.
End.
```

Fig. 2.6 - Algorithms for NN-searching in information retrieval (see text for details).

If $\bar{m}$ and $\bar{n}$ are, respectively, the average numbers of terms associated with a document, and the average number of documents associated with a term, then the average complexity (over all n NN searches) of this usual algorithm is $O(n\,\bar{m})$. It is assumed that advantage is taken of some packed form of storage (i.e. some representation of linked lists) in the inner loop.

Croft's algorithm (see Croft ,1977) appears at first sight to be of complexity $O(n\,m^2)$. However the number of terms associated with the document whose NN is required will often be quite small. The National Physical Laboratory test collection (used in Smeaton and van Rijsbergen, 1981, and Murtagh, 1982), for example, has the following statistics : n = 11429, m = 7491, $\bar{m}$ = 19.9 and $\bar{n}$ = 30.4. The outermost and innermost loops in Croft's algorithm use the document-term file. The centre loop uses the term-document inverted file. The average complexity, over all n NN searches, is easily seen to be $O(n\,\bar{m}^2)$.

In the outermost loop of Croft's algorithm, there will eventually come about a situation where - if a document has not been thus far examined - the number of terms remaining for the given document do not permit the current NN document to be bettered. In this case we can cut short the iterations of the outermost loop. The calculation of a bound, using the greatest possible number of terms which could be shared with a so-far unexamined document has been exploited by Smeaton and van Rijsbergen (1981) and by Murtagh (1982) in successive improvements on Croft's algorithm.

The complexity of all the above algorithms has been measured in terms of operations to be performed. In practice, however, the actual accessing of terms or documents is of far greater cost. The document-term and term-document files must be stored on direct access storage because of their large sizes. Carrying out an I/O operation is very much more costly than any operation

in central memory. Note also that the strategy used in the foregoing algorithms (Croft's algorithm and the above-mentioned improvements on it) does not allow any viable approaches to batching together the records which are to be read successively, in order to improve the accessing performance.

The Perry-Willett algorithm (see Perry and Willett, 1983) presents a simple but effective solution to the problem of costly I/O. It focusses on the calculation of $\#(i \cap i')$, i.e. the number of terms common to the given document i and each other document i' in the collection. This set of values is built up in a computationally efficient fashion. O(n) operations are subsequently required to determine the (dis)similarity, using another vector, $\{\#(i') | i' = 1,2,\ldots,n\}$, stored in main memory. Finally, the best (dis) similarity can be simultaneously determined when carrying out these operations. Hence computation time is $O(\bar{n}\,\bar{m} + n)$. We will now turn attention to the numbers of direct access reads required.

In Croft's algorithm, all terms associated with the document whose NN is desired may be read in one read operation. Subsequently, we require $\bar{n}\,\bar{m}$ reads, giving in all $1 + \bar{n}\,\bar{m}$. In the Perry-Willett algorithm, the outer loop again pertains to the one (given) document, and so all terms associated with this document can be read and stored. Subsequently, $\bar{m}$ reads (i.e. the average number of terms, each of which demands a read of a set of documents) are required, giving in all $1+\bar{m}$. Since these reads are very much the costliest operation in practice, the Perry-Willett algorithm can be recommended for large values of n and m. Its general limitations are that (1) it requires, as do all the algorithms discussed in this section, the availability of the inverted term-document file; and (2) it requires in-core storage of two vectors containing n integer values.

2.7 OPEN PROBLEMS

Approximation algorithms which find a close point, but which is not guaranteed to be a NN, have also been proposed. An exact search algorithm is however required for clustering (see Chapters 3 and 4) and in most other applications. More work is required in formulating general strategies for use with high-dimensional data : as may be noted in the previous sections, most experimentation has been on relatively small dimensions. For incorporation in clustering algorithms, the problem of dynamization is important also. This is where the NN algorithm caters for a dynamic data set, i.e. insertions and deletions to the data set are efficiently carried out.

2.8 REFERENCES.

J.L. BENTLEY, Multidimensional binary search trees in database applications. IEEE Transactions on Software Engineering SE-S, 333-340 (1979).

J.L. BENTLEY, A case study in applied algorithm design. Computer 17, 75-84 (1984).

J.L. BENTLEY and J.H. FRIEDMAN, Fast algorithms for constructing minimal spanning trees in coordinate spaces. IEEE Transactions on Computers C-27, 97-105 (1978).

J.L. BENTLEY, B.W. WEIDE and A.C. YAO, Optimal expected time algorithms for closest point problems. ACM Transactions on Mathematical Software 6, 563-580 (1980).

W.A. BURKHARD and R.M. KELLER, Some approaches to best-match file searching. Communications of the ACM 16, 230-236 (1973).

W.B. CROFT, Clustering large files of documents using the single-link method. Journal of the American Society for Information Science 28, 341-344 (1977).

C. DELANNOY, Un algorithme rapide de recherche de plus proches voisins. RAIRO Informatique/ Computer Science 14, 275-286 (1980).

C.M. EASTMAN and S.F. WEISS, Tree structures for high dimensionality nearest neighbour searching. Information Systems 7, 115-122 (1982).

J.H. FRIEDMAN, F. BASKETT and L.J. SHUSTEK, An algorithm for finding nearest neighbours. IEEE Transactions on Computers C-24, 1000-1006 (1975).

J.H. FRIEDMAN, J.L. BENTLEY and R.A. FINKEL, An algorithm for finding best matches in logarithmic expected time. ACM Transactions on Mathematical Software 3, 209-226 (1977).

K. FUKUNAGA and P.M. NARENDRA , A branch and bound algorithm for computing k-nearest neighbours. IEEE Transactions on Computers C-24, 750-753 (1975).

J. KITTLER, A method for determinining k-nearest neighbours. Kybernetes 7, 313-315 (1978).

R.B. MARIMONT and M.B. SHAPIRO, Nearest neighbour searches and the curse of dimensionality . J. Inst. Maths. Applics. 24, 59-70 (1979).

F. MURTAGH, A very fast, exact nearest neighbour algorithm for use in information retrieval. Information Technology 1, 275-283 (1982).

F. MURTAGH, Expected time complexity results for hierarchic clustering algorithms which use cluster centres. Information Processing Letters 16, 237-241 (1983).

S.A. PERRY and P. WILLETT,A review of the use of inverted files for best match searching in information retrieval systems. Journal of Information Science 6, 59-66 (1983).

M. RICHETIN, G. RIVES and M. NARANJO, Algorithme rapide pour la détermination des k plus proches voisins. RAIRO Informatique/Computer Science 14, 369-378 (1980).

F.J. ROHLF, A probabilistic minimum spanning tree algorithm. Information Processing Letters 7, 44-48 (1978).

M. SHAPIRO, The choice of reference points in best-match file searching. Communications of the ACM 20, 339-343 (1977).

A.F. SMEATON and C.J. VAN RIJSBERGEN, The nearest neighbour problem in information retrieval : an algorithm using upperbounds. ACM SIGIR Forum 16, 83-87 (1981).

S.F. WEISS, A probabilistic algorithm for nearest neighbour searching. In R.N. Oddy et al. (eds.), Information Retrieval Research, Butterworths, London, 325-333 (1981).

T.P. YUNCK, A technique to identify nearest neighbours. IEEE Transactions on Systems, Man, andCybernetics SMC-6, 678-683 (1976).

CHAPTER 3 - SYNOPTIC CLUSTERING

3.1 INTRODUCTION

The problem of synoptic clustering is very different from the search for "natural" clusters or patterns. In the latter problem, the input data must be faithfully respected; in the synoptic clustering problem convenient output is of greater interest.

This Chapter will chiefly focus on Ward's method, or the minimum variance method. For most applications this method can be recommended for the summarization of data. Section 3.2 will attempt to justify this statement : it will informally describe the minimum variance agglomerative method, and will detail properties of this method which are of practical importance.

Mathematical properties of the minimum variance method are detailed in section 3.4. Preceding this, section 3.3 establishes the equivalence of agglomerative algorithms based on distances (or dissimilarities) and algorithms based on cluster centres. The former is the traditional approach, while the latter is used in the recently-proposed, fast algorithms described in later sections. Apart from the minimum variance method, other agglomerative methods which can be implemented with very little alteration are concurrently described. The geometric methods all have the property of defining a cluster centre, or cluster representative.

An essential property of the fast algorithms - a precondition for their use, which depends on the cluster criterion - is discussed in section 3.5. Sections 3.6 and 3.7 describe the fast algorithms, and prove the major performance results.

3.2 MINIMUM VARIANCE METHOD IN PERSPECTIVE

Agglomerative clustering methods have been motivated by graph theory - leading to linkage-based methods - or by geometry. This generalization, which is true for the more commonly used methods, indicates the most productive frameworks for studying clustering methods. In geometric methods, the cluster centre may be used for subsequent agglomerations. Alternatively, inter-cluster dissimilarities may be used throughout, and therefore these methods may be implemented using the Lance-Williams dissimilarity update formula (see Table 1, section 3.3 below) as in the case of linkage-based methods.

In order to specify an agglomerative criterion simultaneously in terms of cluster mean vectors, and in terms of dissimilarity, it is necessary to adopt a particular dissimilarity. In this Chapter, it will be assumed that the Euclidean distance is employed. Restricting the choice of dissimilarity to this distance is rarely inconvenient in practice. The Euclidean distance is often the most natural choice of distance, and a Euclidean space offers a well-known and powerful standpoint for analysis.

The variance or spread of a set of points (i.e. the sum of squared distances from the centre) has been the point of departure for specifying many clustering algorithms. Many of these algorithms, - iterative, optimization algorithms as well as the hierarchical, agglomerative algorithm - are briefly described and appraised in Wishart (1969). The use of variance in a clustering criterion links the resulting clustering to other data-analytic techniques which involve a decomposition of variance. Principal components analysis, for example, seeks the principal directions of elongation of the multidimensional points, i.e. the axes on which the projections of the points have maximal variance. Using a clustering of the points with minimal variance within clusters as the cluster-criterion is, perhaps, the most suitable criterion for two concurrent but compli-

mentary analyses of the same set of points. The reality of clusters of projected points resulting from the principal components analysis may be assessed using the cluster analysis results; and the interpretation of the axes of the former technique may be used to facilitate interpretation of the clustering results.

The search for clusters of maximum homogeneity leads to the minimum variance criterion. Since no coordinate axis is privileged by the Euclidean distance, the resulting clusters will be approximately hyperspherical. Such ball-shaped clusters will therefore be very unsuitable for examining straggly patterns of points. However, in the absence of information about such patterns in the data, homogeneous clusters will provide the most useful condensation of the data.

The following properties make the minimum variance agglomerative strategy particularly suitable for synoptic clustering.

(1) As discussed in section 3.4, the two properties of cluster homogeneity and cluster separability are incorporated in the cluster criterion. For summarizing data, it is unlikely that more suitable criteria could be devised.

(2) As in the case of other geometric strategies, the minimum variance method defines a cluster centre of gravity. This mean set of cluster members' coordinate values is the most useful summary of the cluster. It may also be used for the fast selection and retrieval of data, by matching on these cluster representative vectors rather than on each individual object vector.

(3) A top-down hierarchy traversal algorithm may also be implemented for information retrieval. Using a query vector, the left or right subtree is selected at each node for continuation of the traversal (-it is best to ensure that each node has precisely two successor nodes in the construction of the hierarchy).

Such an algorithm will work best if all top-down traversals through the hierarchy are of approximately equal length. This will be the case if and only if the hierarchy is as "symmetric" or "balanced" as possible (see Fig. 3.1). Such a balanced hierarchy is usually of greatest interest for interpretative purposes also : a partition, derived from a hierarchy, and consisting of a large number of small classes, and one or a few large classes, is less likely to be of practical use.

For such reasons a "symmetric" hierarchy is desirable. It has been shown, using a number of different measures of hierarchic symmetry, that the minimum variance (closely followed by the complete link) methods generally give the most symmetric hierarchies (see Murtagh, 1984). This is an important, albeit "post festum ", property of hierarchies produced by the minimum variance method.

(4) Unlike other geometric agglomerative methods - in particular the centroid and the median methods (see definitions, Table 1, below) - the sequence of agglomerations in the minimum variance method is guaranteed not to allow inversions in the cluster criterion value. Inversions or reversals (Fig. 3.2) are inconvenient, can make interpretation of the hierarchy very difficult, and will be investigated in the following section.

(5) Finally, computational performance has until recently favoured linkage-based agglomerative criteria, and in particular the single linkage method. The computational advances described below for the minimum variance method (and other methods) make it increasingly attractive for practical applications involving large amounts of data. Some of the algorithms to be discussed lend themselves to a parallel implementation : as parallel machine architectures become more widely available, research in this direction should be particularly fruitful.

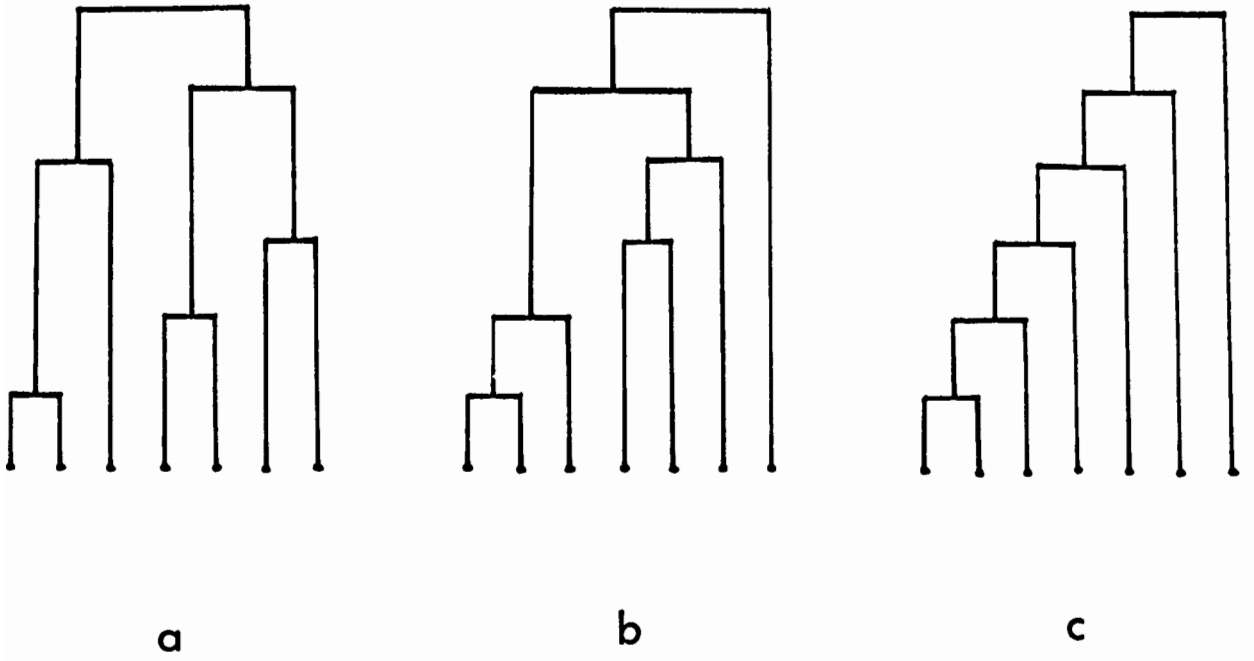


Fig. 3.1 - Three binary hierarchies with n=7 terminal or leaf nodes; (a) is symmetric, (c) is as asymmetric as possible and (b) presents an intermediate case.

In this Chapter, references to many of the original proposals for the algorithms described may be consulted in Murtagh (1983).

Fig. 3.2 - Alternative representations of a hierarchy with an inversion (reversal) in cluster criterion values.

3.3 GEOMETRIC AGGLOMERATIVE METHODS

Hierarchic agglomerative algorithms may be conveniently broken down into linkage (or graph theoretic) methods - the single, complete, weighted and unweighted average linkage methods - and cluster centre (or geometric) methods - the centroid, median, and minimum variance methods. The latter may be specified either in terms of dissimilarities, alone, or alternatively in terms of cluster centre coordinates and distances (dissimilarities, in the case of the minimum variance method). Let us prove this in the case of the median method.

Let a and b be two points (i.e. m-dimensional vectors: objects or cluster centres) which have been agglomerated, and let c be another point. From the Lance-Williams dissimilarity update formula, using squared Euclidean distances, we have :

$$\begin{aligned} d^2(a\cup b,c) &= d^2(a,c)/2 + d^2(b,c)/2 - d^2(a,b)/4 \\ &= (a-c)^2 + (b-c)^2/2 - (a-b)^2/4 \end{aligned} \qquad (1)$$

The new cluster centre is (a+b)/2, so that its distance to point c is

$$(c- (a+b)/2)^2 \qquad (2)$$

That these two expressions are identical is readily verified. The correspondence between these two perspectives on the one agglomerative criterion is similarly proved for the centroid and minimum variance methods.

For these geometric methods, and with suitable alterations for graph methods, the following algorithm is an alternative to the general dissimilarity-based algorithm described in Chapter 1 (which may be described as a "stored dissimilarities approach").

Hierarchical clustering methods (and aliases).	Lance and Williams dissimilarity update formula.	Coordinates of centre of cluster, which agglomerates clusters i and j.	Dissimilarity between cluster centres g_i and g_j.
Single link (nearest neighbour).	a(i)=0.5 b=0 c=-0.5 (More simply: min{d(i,k),d(j,k)}.)	-	-
Complete link (diameter).	a(i)=0.5 b=0 c=0.5 (More simply: max{d(i,k),d(j,k)}.)	-	-
Group average (average link, UPGMA).	a(i)=|i|/(|i|+|j|) b=0 c=0	-	-
McQuitty's method (WPGMA).	a(i)=0.5 b=0 c=0	-	-
Median (Gower's method, WPGMC).	a(i)=0.5 b=-0.25 c=0	$g=(g_i+g_j)/2$	$\|\|g_i-g_j\|\|^2$
Centroid (UPGMC).	a(i)=|i|/(|i|+|j|) $b=-\|i\|.\|j\|/(\|i\|+\|j\|)^2$ c=0	$g=(\|i\|g_i+\|j\|g_j)/(\|i\|+\|j\|)$	$\|\|g_i-g_j\|\|^2$
Ward's method (minimum variance, error sum of squares).	a(i)=(|i|+|k|)/(|i|+|j|+|k|) b=-|k|/(|i|+|j|+|k|) c=0	$g=(\|i\|g_i+\|j\|g_j)/(\|i\|+\|j\|)$	$(\|i\|.\|j\|/(\|i\|+\|j\|)).\|\|g_i-g_j\|\|^2$

Notes : |i| = number of objects in cluster i.

g_i is a vector in m-space (where m is set of variables); either initial point or cluster centre.

||.|| is the norm in the Euclidean metric.

Lance and Williams recurrence formula ([ ]: absolute difference):

$d(i \cup j,k)=a(i).d(i,k)+a(j).d(j,k)+b.d(i,j)+c.[d(i,k)-d(j,k)]$.

Algorithm A. Stored Data Approach

Step 1 : Examine all interpoint dissimilarities, and form cluster from two closest points.

Step 2 : Replace two points clustered by representative point (centre of gravity) or by cluster fragment.

Step 3 : Return to Step 1, treating clusters as well as remaining objects, until all objects are in one cluster.

In Steps 1 and 2, "point" refers either to objects or clusters, both of which are defined as vectors in the case of geometric methods. This algorithm is justified by storage considerations, since we have O(n) storage required for n initial objects and O(n) storage for the n-1 (at most) clusters. In the case of graph methods, the term "fragment" in Step 2 refers to a connected component in the case of the single link method and to a clique or complete subgraph in the case of the complete link method. The overall complexity of the above algorithm is at best $O(n^3)$: the repeated calculation of dissimilarities in Step 1, coupled with O(n) iterations through Steps 1, 2 and 3. Note however that this does not take into consideration the extra processing required in a graph method, where "closest" in Step 1 is defined with respect to graph fragments.

Apart from lessened storage requirements, Algorithm A can also be justified in that it can be made computationally very efficient by searching for the clusters, in Step 1, in a restricted, local region only. This will be looked at below.

3.4 MINIMUM VARIANCE METHOD : MATHEMATICAL PROPERTIES

The minimum variance method produces clusters which satisfy compactness and isolation criteria. These criteria are incorporated into the dissimilarity, noted in Table 1, as will now be shown.

In Ward's method, we seek to agglomerate two clusters, c_1 and c_2, into cluster c such that the within-class variance of the partition thereby obtained is minimum. Alternatively, the between-class variance of the partition obtained is to be maximized (see Lemma 1, below). Let P and P* be the partitions prior to, and subsequent to, the agglomeration; let p_1, p_2, ... be classes of the partitions:

$$P = \{ p_1, p_2, \ldots, p_k, c_1, c_2 \}$$

$$P^* = \{ p_1, p_2, \ldots, p_k, c \} .$$

Finally, let i denote any individual or object, and I the set of such objects. In the following, classes (i.e. p or c) and individuals (i.e. i) will be considered as vectors or as sets: the context will be sufficient to make clear which is the case. Total variance of the cloud of objects in m-dimensional space is decomposed into the sum of within-class variance and between-class variance. This is Huygen's theorem in classical mechanics (for proof, see Lemma 1 below), and for the two partitions we have respectively:

$$\mathrm{Var}(I) = \mathrm{Var}(P) + \sum_{p \in P} \mathrm{Var}(p)$$

$$\mathrm{Var}(I) = \mathrm{Var}(P^*) + \sum_{p \in P^*} \mathrm{Var}(p)$$

Hence :

$$\mathrm{Var}(P) + \mathrm{Var}(p_1) + \ldots + \mathrm{Var}(p_k) + \mathrm{Var}(c_1) + \mathrm{Var}(c_2)$$
$$= \mathrm{Var}(P^*) + \mathrm{Var}(p_1) + \ldots + \mathrm{Var}(p_k) + \mathrm{Var}(c) \quad .$$

Therefore :

$$\mathrm{Var}(P^*) = \mathrm{Var}(P) + \mathrm{Var}(c_1) + \mathrm{Var}(c_2) - \mathrm{Var}(c) \quad .$$

In agglomerating two classes of P, the variance of the resulting partition (i.e. Var(P*)) will necessarily decrease: therefore in seeking to minimize this decrease, we simultaneously achieve a partition with maximum between-class variance. The criterion to be optimized is then :

$$\mathrm{Var}(P) - \mathrm{Var}(P^*)$$
$$= \mathrm{Var}(c) - \mathrm{Var}(c_1) - \mathrm{Var}(c_2)$$
$$= \frac{|c_1| \, . \, |c_2|}{|c_1| + |c_2|} \; ||c_1 - c_2||^2 \qquad \text{(see Lemma 2 below),}$$

which is the dissimilarity given in Table 1. This is a dissimilarity which may be determined for any pair of classes of partition P; and the agglomerands are those classes, c_1 and c_2, for which it is minimum. Having shown what we set out to prove, we now turn attention to two results which were used in the foregoing.

Lemma 1 : T = W + B, where T is the total variance of the set of points, I; W is the within-class variance of a partition of I; and B is the between-class variance of this partition.

Proof : Let partition P be defined as $\{p_1, p_2, \ldots, p_k\}$.
We are to prove that :

$$\mathrm{Var}(I) = \mathrm{Var}(P) + \sum_{p \in P} \mathrm{Var}(p)$$

i.e.

$$\frac{1}{n}\sum_{i\in I}(i-g)^2 = \sum_{p\in P}\frac{|p|}{n}(p-g)^2 + \frac{1}{n}\sum_{p\in P}\sum_{i\in p}(i-p)^2$$

where g is the grand mean of the n objects : $g = \frac{1}{n}\sum_{i\in I} i$; and $|p|$ is the cardinality of class p. Note that p is used interchangeably to denote centre of gravity (a vector), and the set whose centre of gravity this is.

Rewriting the rightmost term in the above expression gives us :

$$\sum_{p\in P} Var(p) = \frac{1}{n}\sum_{p\in P}\sum_{i\in p}(i-p)^2 = \sum_{i\in I}\frac{1}{n}((i-g)-(p-g))^2$$

$$= \sum_{i\in I}\frac{1}{n}(i-g)^2 + \sum_{i\in I}\frac{1}{n}(p-g)^2 - 2\sum_{i\in I}\frac{1}{n}(i-g)(p-g)$$

$$= \frac{1}{n}\sum_{i\in I}(i-g)^2 + \sum_{p\in P}\frac{|p|}{n}(p-g)^2 - \frac{2}{n}\sum_{p\in P}|p|(p-g)^2$$

$$= \frac{1}{n}\sum_{i\in I}(i-g)^2 - \sum_{p\in P}\frac{|p|}{n}(p-g)^2$$

$$= Var(I) - Var(P) .$$

Corollary: For any choice of P, T is constant; therefore any choice of P which maximizes W (a measure of class compactness) simultaneously minimizes B (a measure of class isolation).

Lemma 2: $Var(c) - Var(c_1) - Var(c_2) = \frac{|c_1|.|c_2|}{|c_1|+|c_2|}\ ||c_1 - c_2||^2$

where $c = c_1 \cup c_2$.

Proof: Consider the variance of class c which is decomposed into classes c_1 and c_2. By Lemma 1 we have:

$$\mathrm{Var}(c) = \mathrm{Var}(\{c_1,c_2\}) + \mathrm{Var}(c_1) + \mathrm{Var}(c_2).$$

Hence, we are to show that

$$\mathrm{Var}(\{c_1,c_2\}) = \frac{|c_1|.|c_2|}{|c_1|+|c_2|} \; ||c_1 - c_2||^2$$

Since c is the centre of gravity of vectors c_1 and c_2 we have :

$$\mathrm{Var}(\{c_1,c_2\}) = |c_1| \; . \; ||c_1-c||^2 \; + \; |c_2|.||c_2-c||^2$$

Writing

$$c = \frac{|c_1|.c_1 + |c_2|.c_2}{|c_1| + |c_2|}$$

and substituting gives the desired result.

Corollary: If c_1 and c_2 are singleton classes, then $\mathrm{Var}(\{c_1,c_2\})$ $= \frac{1}{2} ||c_1 - c_2||^2$ (i.e. the Euclidean distance between a pair of classes is twice their variance).

3.5 REDUCIBILITY PROPERTY

The reducibility property lays down the condition under which the fast algorithms, reviewed in subsequent sections, may be constructed locally, by carrying out agglomerations in restricted regions of the space of objects, but such that the hierarchy arrived at is nonetheless exact. This reducibility property is also closely related to the problem of reversals or inversions in hierarchic, agglomerative algorithms, i.e. $d(a \cup b, c) \not\geqslant d(a, b)$ for some clusters or objects a, b, and c and where a and b agglomerate to constitute $a \cup b$ (see Fig. 3.2). Following the statement of the reducibility property, three other equivalent or derived expressions of this property will be examined. The importance of this property for clustering algorithms will then be studied.

Reversals and the reducibility property.

Consider the agglomeration of clusters or objects, a and b, into $c = a \cup b$. Consider also some other cluster or object c'. The reducibility property is then:

$$d(a,b) \leqslant \inf\{d(a,c'), d(b,c')\}$$
$$\Rightarrow \inf\{d(a,c'), d(b,c')\} \leqslant d(a \cup b, c'). \qquad (3)$$

Verbally, the agglomeration of a and b cannot produce a cluster $c = a \cup b$ which will be closer to cluster c' than was either a or b. If such were the case, then it is possible that c and c' might subsequently agglomerate at a smaller proximity than in the case of the agglomeration of a and b: i.e. there would be an inversion or reversal. Thus, if a clustering strategy satisfies the reducibility property, inversions cannot arise. This is proved, for any given clustering strategy, by using the Lance-Williams dissimilarity update formula. Before looking at Ward's method as an example, it will be convenient to derive an equivalent ex-

pression of the reducibility property.

If $d(a,b) \leqslant \rho \leqslant \inf \{ d(a,c'), d(b,c') \}$, then from (3) we have: $\rho \leqslant \inf \{ d(a,c'), d(b,c') \} \leqslant d(c,c')$. Therefore we can write:

$$\left.\begin{array}{l} d(a,b) \leqslant \rho \\ d(a,c') \geqslant \rho \\ d(b,c') \geqslant \rho \end{array}\right\} \Rightarrow d(c,c') \geqslant \rho \qquad (4)$$

This form of the reducibility property allows easy verification. Taking Ward's method, we have:

$$d(c,c') = \left((|a|+|c'|)d(a,c') + (|b|+|c'|)d(b,c') - |c'|\, d(a,b) \right) / \left(|c|+|c'| \right)$$

Using (4), the right hand side is

$$\geqslant \rho \cdot \left((|a|+|b| + |c'| \right) / \left(|c|+|c'| \right)$$

which is necessarily $\geqslant \rho$. Therefore $d(c,c')$ is also $\geqslant \rho$. The proof that the single, complete, and average linkage methods satisfy the reducibility property is similarly proved. In the case of the centroid and median methods, however, it is not possible to guarantee that $d(c,c') \geqslant \rho$.

Two further expressions for the reducibility property will be derived. In (3) consider some ρ greater than or equal to the terms on both sides of the implication. We have:

$$\begin{array}{l} d(a,b) \leqslant \rho \\ d(c,c') \leqslant \rho \end{array}$$

and we must necessarily have:

$$d(a,c') \leqslant \rho \quad \text{or} \quad d(b,c') \leqslant \rho \tag{5}$$

Since the contrary does not necessarily hold, if we are using a geometric method where each of a, b, and c are reduced to a point, we have that

$$B_\rho(c) \subseteq B_\rho(a) \cup B_\rho(b) \tag{6}$$

where $B_\rho(\ )$ is the hypersphere of specified centre and of radius ρ. More generally, B_ρ may be defined as the set of all possible points within a dissimilarity ρ of the cluster members. Because of the latter expression, the reducibility property may also be referred to as a space contraction property. Note that since the reducibility property is satisfied by the complete link method, the property of spatial contraction is used here in a different sense to its use by Lance and Williams (1967).

Reciprocal nearest neighbours and the reducibility property.

Let the nearest neighbour graph (NN-graph) be defined as a set of points, p, whose directed edges {(p,NN(p))} are such that NN(p) is the nearest neighbour of p. For any point p, three cases can be distinguished: see Fig. 3.3. In case III, where q = NN(p), and p = NN(q), points p and q are referred to as mutual or reciprocal nearest neighbours (RNNs).

The NN-graph can be defined at all stages of the construction of the hierarchy for a geometric clustering method, - where the vertices consist of remaining, unclustered, initial points and cluster representative points - and with suitable additional operations can also be used for a graph clustering method. Before looking at algorithms for these different areas, consider first the initial situation where the set of n multi-

dimensional points are given. If the reducibility porperty is verified by a clustering method, the merging of RNNs i and j into cluster $i \cup j$ requires the updating of the NN-graph only for those points which had i or j as NN (cf. Case II, Fig. 3.3). More importantly it means that all RNNs i and j can be simultaneously merged, without effecting the RNN properties of other parts of the NN-graph.

These two corollaries of the reducibility property will be used in the algorithms to be studied in the next 2 sections.

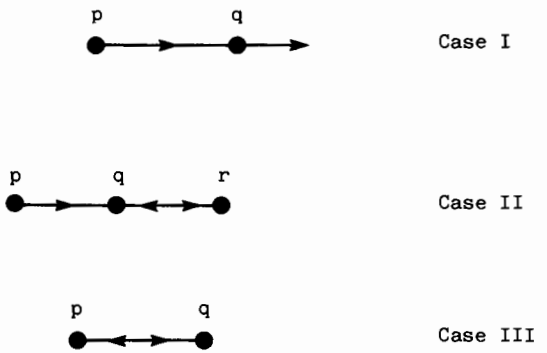


Fig. 3.3 - Three situations of interest in the NN-graph.

3.6 MULTIPLE CLUSTER ALGORITHM

The following algorithm (Algorithm B) provides $O(n^2)$ worst case time and $O(n)$ space results for geometric strategies which have the additional property (as will be seen following Algorithm B) of defining inter-cluster dissimilarity as distance. From Table 1 (3rd column), this is the case for the centroid and median methods but not for Ward's method. These complexity results therefore will apply to the approximate centroid and median hierarchies obtained using Algorithm B. (The approximation relates to the non-verification of the reducibility property by these methods. For exact results, RNNs would need to be considered in decreasing order of closeness, in order to allow for the effects produced by an agglomeration on other RNNs. Instead, for computational efficiency, the RNNs will be processed here in the order that they are found. Discrepancy with exact algorithms for the centroid and median methods will be obtained when there is a reversal or inversion, and will rarely be inconvenient from a practical point of view.)

Algorithm B. Parallel clustering

Step 1. (Re)determine all NNs and RNNs.

Step 2. Agglomerate all RNNs, replacing with cluster point.

Step 3. Go to Step 1 until one point remains.

Let t be the number of iterations carried out ($t \leqslant n-1$). On the first execution of Step 2, if the number of RNNs found is r_1, then r_1 agglomerations are carried out, and $n-r_1$ points remain. We have: $1 \leqslant r_1 \leqslant \lfloor n/2 \rfloor$, since at least one point, and at most half the point-set, will be RNNs. Since there are precisely $n-1$ agglomerations, if we find on successive iterations $r_2, r_3, \ldots, r_t$ RNNs in Step 2, we have:

$$r_1 + r_2 + \ldots\ldots + r_t = n-1 \qquad (7)$$

Good time complexity results for this algorithm depend (as will be seen) on whether or not the number of NN calculations to be carried out following any agglomeration is constant.

Complexity: $O(n^2)$ computation and $O(n)$ storage for approximate median and centroid methods.

Proof: In Step 1, n dissimilarity calculations are carried out, each of which requires n-1 calculations. The RNNs are determined in $O(n)$ operations, by a straightforward scan of the n NNs. In Step 2, r_1 RNNs are found. Let the number of points p corresponding to Case II of Fig. 3.3 be upper-bounded by a constant for each such pair of RNNs: this supposition is discussed below. Then there will be less than or equal to $r_1 + c.r_1$ new NN calculations to be performed (i.e the NNs of the r_1 new cluster centres will be required, as will the new NNs of the - at most- c/2 points which correspond to point p of Case II in Fig. 3.3). Similarly, on the following iteration, there will be at most $r_2 + c.r_2$ new NNs to be determined. Thus, in all, the number of NN calculations is bounded from above by

$$
\begin{aligned}
& n + r_1(c+1) + r_2(c+1) + \ldots + r_t(c+1) \\
&= n + (c+1)(n-1) && \text{using (7)} \\
&= O(n)
\end{aligned}
$$

Each such calculation will take less than or equal to n-1 dissimilarity calculations, giving overall complexity of $O(n^2)$.

Proof of supposition: The number of points which can have a given point as NN in the Euclidean plane is less than or equal to 6 (consider the limit case where the given point is placed at the centre of a circle of radius ρ, on whose perimeter the 6 points are placed such that all ad-

jacent pairs are precisely a distance ρ apart). In higher dimensional spaces, use may be made of the fact that the Euclidean distance is upper-bounded by the Chebyshev (∞ or maximum coordinate) distance. In the L_∞ metric, hyperspheres of radius ρ become (from an L_2 perspective) cubes of edge length 2ρ. The number of points which can simultaneously have a given point as NN in m-dimensional Chebyshev space is then 3^m-1 (i.e. the number of cubes which are adjacent to a given cube). For given m, this is constant. Although we may be reasonably happy that this lower bound holds also for the Euclidean distance, it is a great deal more difficult to prove this. Day and Edelsbrunner (1984) may be consulted for this proof.

Unfortunately the above result does not hold for Ward's method, where a dissimilarity between cluster centres results from each agglomeration (cf. column 3, Table 1). A worst case of O(n) NN calculations must be assumed to follow each agglomeration, which leads to overall $O(n^3)$ worst case time.

Due to the fact that each agglomeration leaves certain points unmatched (cf. Case II, Fig. 3.3), Algorithm B has been dubbed "algorithme des célibataires" in the Vol. VII, No. 2, 1982 issue of the journal Les Cahiers de l'Analyse des Données which focussed on these new approaches to clustering using NNs, RNNs, and NN-chains. Because the number of célibataires after each batch of agglomerations increases exponentially with the dimensionality of the point-space, it appears that Algorithm B is only feasible for relatively low dimensional spaces (about 4 or 5 at most). Algorithm C, to be discussed in the next section, is more practicable. Algorithm B, however, allows agglomerations to be carried out in as parallel a fashion as possible and thus may well be a natural choice of algorithm as parallel processors become more widely available.

An exact version of this algorithm for the centroid and median methods may be obtained with a small amount of extra processing. The smallest distance is found at each agglomeration. This will require only O(n) extra processing, given the list of NNs, and the smallest dissimilarity is necessarily a RNN pair. The agglomeration is carried out, and the list of NNs updated in O(n) time. There are n-1 iterations, and hence the overall time complexity remains $O(n^2)$. Algorithm B differs in that the RNN pair of least dissimilarity is not obtained (thus saving O(n) calculations on each iteration); and that all updates of the list of NNs can be carried out on the reduced set of clusters/objects which follows each batch of agglomerations (thus saving further time).

3.7 SINGLE CLUSTER ALGORITHM

Rather than carrying out as many agglomerations as possible at each iteration, the algorithms to be discussed in this section only carry out one agglomeration per iteration.

The single cluster algorithm is based on the NN-chain. Starting with an arbitrary object or cluster, i, this is defined as

$$i, NN(i)=j, NN(j)=k, \ldots, NN(p)=q, NN(q)=p.$$

Let it be assumed that no two dissimilarities are equal (arbitrary resolving of such cases will be discussed below). Then the following three propositions hold for NN-chains:

Proposition 1 : Inter-object/cluster dissimilarities monotonically decrease as we proceed along the NN-chain.

Proposition 2 : The final link always connects a RNN pair.

Proposition 3 : The NN-chain cannot contain a circuit of more than two nodes.

Proof 1 : If we have $NN(i)=j$ and $NN(j)=k$, for $i \neq k$, then necessarily: $d(i,j) > d(j,k)$, since otherwise i would be the NN of j, contrary to construction.

Proof 2 : For some p, in determining $q=NN(p)$, we must have either $q \notin$ NN-chain, in which case a new link is grown onto the NN-chain; or $q \in$ NN-chain; by Proposition 1, this can only be the object/cluster which

precedes p in the NN-chain.

Proof 3 : If we had a cycle, for some i, j, ... , p, q:

i, NN(i)=j, ... , NN(p)=q, NN(q)=i

then Proposition 1 would be violated.

Owing to Proposition 1, we may say that the NN-chain is grown towards increasing density, since inter-point dissimilarity - hence sparseness - at the start of the NN-chain is greater than that at the end.

In practice, some dissimilarities might be equal. In any implementation of the algorithm to be described below, arbitrary resolution of such cases must be provided for. In particular, a circuit such as

i, NN(i)=j, NN(j)=k, NN(k)=i

where d(i,j) = d(j,k) = d(k,i)

must be prevented in the NN-chain.

Algorithm C, as follows, is suitable for any geometric strategy.

Algorithm C. NN-chain clustering

Step 1. Select a point arbitrarily.

Step 2. Grow the NN-chain from this point until a pair of RNNs are obtained.

Step 3. Agglomerate these points, replacing with a cluster point.

Step 4. From the point which preceded the RNNs, or from an arbitrary point if there is no such point, go to Step 2 until only one point remains.

If i is the first point selected, we obtain the sequence

i, NN(i)=j, NN(j)=k, ... , NN(o)=p, NN(p)=q, NN(q)=p

(Note that i and j could constitute a RNN pair). Points p and q are merged, and a new point replaces them. This contraction of the NN-chain is followed by a further set of growths starting from point o (or from an arbitrary point if the RNN pair were the only two points in the NN-chain). Algorithm C is exact if agglomeration of a RNN pair doesn't affect the RNN properties of any other objects and clusters, i.e. if the reducibility property is satisfied by the agglomerative strategy used.

Complexity: Algorithm C is optimal for all geometric methods; i.e. it requires $O(n^2)$ computation and $O(n)$ storage.

Proof: Let a growth of the NN-chain refer to the adding of a link, and a contraction refer to the agglomeration of a pair of RNNs. Algorithm C is seen to be a series of intermixed growths and contractions. The number of contractions is n-1 (i.e. the number of agglomerations). The number of growths cannot exceed 3n-3: i.e. the number of nodes incorporated into the NN-chain can never exceed the n initial points, plus the n-1 cluster points created, which gives a total of 2n-2 links; and a final set of n-1 links must be considered which allow a RNN pair to be made out of the final link in the NN-chain. Now, each growth requires 1 NN calculation. Each contraction requires a constant number of operations. Therefore, the overall complexity - assuming $O(n)$ effort for a NN calculation - is $O(n^2)$. Storage of the NN-chain, the original data, and the cluster points, is altogether $O(n)$.

In growing a link onto a NN-chain, when n_1 points are in the NN-chain and

n_2 points are not ($n_1+n_2 \leqslant n$), the number of NN calculations to be carried out is n_2+1: dissimilarities between the last point in the NN-chain and the n_2 points not in the chain must be determined, as also the dissimilarity between the last and the second last points in the NN-chain. It is fruitless to examine dissimilarities with any of the other n_1 points since these must be greater that the NN dissimilarity required.

For graph (or linkage) methods, where inter-cluster dissimilarity cannot be calculated in O(1) time unless we have the entire set of dissimilarities directly accessable, Algorithm C may be amended as follows.

Algorithm D. NN-chain algorithm using stored dissimilarities.

Step 1. Select an object arbitrarily.

Step 2. Determine and store all inter-object dissimilarities.

Step 3. Grow the NN-chain from the object chosen, until a pair of RNNs are obtained.

Step 4. Agglomerate these objects.

Step 5. Update the dissimilarity table, using the Lance-Williams formula.

Step 6. From the node in the NN-chain which preceded the RNNs, or from an arbitrary node (object or cluster) if the NN-chain is empty, go to Step 3 until only one node remains.

Clearly, $O(n^2)$ storage is required here. As before, there are O(n) growths of the NN-chain, each requiring O(n) updating of the dissimilarity table. In total, therefore, computational complexity is $O(n^2)$.

Algorithm D provides time-optimal algorithms for the weighted and unweighted average linkage methods (UPGMA, WPGMA). It does so also for any general strategy based on the Lance-Williams recurrence formula, and will be exact if the agglomerative strategy satisfies the reducibility property.

It does not appear that the $O(n^2)$ computational complexity of Algorithm D can be further improved. However, the complexity of Algorithm B is seen to be $O(nf)$ where $O(f)$ is the complexity of finding a nearest neighbour of a point and a brute-force approach to this subproblem is $O(n)$. Efficient linear and sub-linear algorithms can instead be used to obtain nearest neighbours. Such algorithms have been reviewed in Chapter 2.

3.8 REFERENCES

W.H.E. DAY and H. EDELSBRUNNER, Efficient algorithms for agglomerative hierarchical methods. Journal of classification 1, 7-24, 1984.

G.N. LANCE and W.T. WILLIAMS, A general theory of classificatory sorting strategies. I. Hierarchical systems. The Computer Journal 9, 373-380, 1967.

F. MURTAGH, A survey of recent advances in hierarchical clustering algorithms. The Computer Journal 26, 354-359, 1983.

F. MURTAGH, Structures of hierarchic clusterings: implications for information retrieval and for multivariate data analysis. Information Processing and Management (in press, 1984).

CHAPTER 4 - CONNECTIVITY CLUSTERING

4.1 INTRODUCTION

Connectivity clustering is particularly important in pattern recognition; and it immediately generalizes to spaces of dimension greater than 2. Rather than attempting to find "useful" clusters, as in the synoptic clustering problem, instead "intuitive" or "natural" patterns are to be analyzed.

Unlike the algorithms discussed in Chapter 3, the algorithms of this Chapter do not require a distance; any dissimilarity may be employed. As before, possible parallel implementations are discussed, and these will probably play an increasing role as new machine architectures become more widespread.

Section 4.2 focusses on the algorithms described in this Chapter, - the single link hierarchical clustering method and the minimal spanning tree (MST). This section gives an indication of where these two closely related methods may be most fruitfully used.

Section 4.3 gives background material on implementations of algorithms. It does not attempt to detail all algorithms which could be employed, but those that it does discuss are among the most efficient and subsequent sections will improve on the basic ideas behind them.

Section 4.4 details implementations of algorithms which may incorporate fast nearest neighbour searching algorithms (see Chapter 2). For general purpose applications, the algorithms of this section could probably be said to represent the most recommendable algorithms when efficiency is of paramount importance. Although the minimal spanning tree is focussed on in this Chapter, the final algorithm of section 4.4 is suitable for any graph (or linkage) method, - such as for example the complete link method.

Since the MST and single linkage methods work on dissimilarities, we may further consider the case when only some of the n(n-1)/2 possible (symmetric) dissimilarities are presented to the algorithm. The problem of such sparse graphs arises, for example, in constrained clustering (to be described in Chapter 5). Section 4.5 details an efficient algorithm for the sparse graph problem. It further discusses a special case of this problem when the graph represents a set of planar points or objects (i.e. the graph is said to be planar).

As with all clustering techniques, there are limits to the applicability of the MST or single linkage method for distinguishing patterns. One way to extend their applicability is to use information provided by the problem in addition to the basic idea of connectivity. Section 4.6 discusses a number of algorithms of this type, where the objective is to find modes or peaks in point density or some analogous measure (irrespective of pattern shape).

4.2 SINGLE LINK METHOD IN PERSPECTIVE.

Hierarchical clustering methods based on the geometric paradigm have been explored in Chapter 3. In this Chapter, methods based on graph-theoretic principles will be studied.

The general hierarchical clustering algorithm described in Chapter 1 allows $O(n^2)$ time and $O(n^2)$ space implementations of the single and complete linkage methods, and of the weighted and unweighted average linkage methods (see Table 1, Chapter 3 for the dissimilarity update formulas required for these methods). At all stages of the agglomerations, the results obtained so far may be graphically presented, without the need to plot cluster centres (see Sneath and Sokal, 1973, for a number of worked examples).

The single link method uses a very weak requirement for cluster formation. The complete link method is more restrictive, and produces hierarchies which are nearly as "balanced" as the minimum variance method (cf. section 3.2). An average linkage method might be preferred for noisy data, so that spurious clusters which might be produced by the over-lax single link method or the over-demanding complete link method can be avoided.

The single link method will be focussed on in this Chapter. Besides being the oldest hierarchical clustering method (it was initially used in the early 1950s), and one of the most widely-used hierarchical methods because of computationally efficient algorithms, it is also of great interest for point pattern recognition. A very wide range of algorithms have, in fact, been developed for the single linkage method: Rohlf (1982) reviews algorithms with complexities ranging from $O(n \log n)$ to $O(n^5)$. Many of these algorithms have first constructed the MST, and subsequently transformed this into the single link hierarchy.

These two problems - single linkage clustering and the MST - are closely related. Information is lost in transforming the MST into the hierarchy, so that the reverse transformation is not possible. Rohlf (1973) describes an efficient $O(n^2)$ worst case algorithm for transforming the MST into the hierarchy. It may be simply sufficient to sort the n-1 edges of the MST, thus providing a representation of the hierarchy (i.e. the sequence of agglomerations; it might be preferred to adopt a clustering labelling standard, such as to number clusters from n+1 to at most 2n-1; or to label clusters by the lowest index number of their object-members). To do this requires $O(n \log n)$ time.

The algorithms described in this Chapter will be based on the MST. Being an important structure itself, and being easily transformable into the hierarchic representation, both indicate the centrality of the MST from a practical stand-point.

The following properties apply to the single link hierarchical method, and to its associated minimal spanning tree.

(1) A number of authors (see for example Jardine and Sibson, 1971) have found the mathematical properties of this method so appealing that they have preferred it to all other hierarchical methods. Among properties not shared by other methods which they have pointed to are:
   - every partition has classes which are optimal with reference to the connectivity criterion used; partitions obtained from the minimum variance method, in contrast, are suboptimal with respect to within-cluster minimum variance.
   - Monotonic transformation of input dissimilarities (i.e. a transformation which preserves order) has no effect on the hierarchy; this may be of importance where a question is raised over the scaling of directly-constructed dissimilarities.

- Small changes in input dissimilarities produce small changes in the hierarchy produced, and thus the single link method is a stable method.

(2) In practice, the single link method has the chaining disadvantage which makes it particularly unsuitable for synoptic clustering.

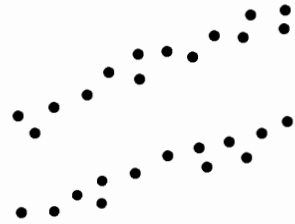
(3) However it is easy to graphically represent the results of linkage based clustering methods on 2-dimensional data; and the hierarchical method or the minimal spanning tree are very suitable for many types of pattern recognition problems.

We will briefly look at problems in this latter area.

Machine vision includes pattern recognition and image processing. The automatic recognition of groups of points is an important problem in the former area. Such point patterns may be arrived at in different ways, - for example, by being derived from a digitized image with a considerably more complex background structure.

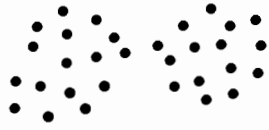
Practically all proposed clustering algorithms would perform well when presented with well-separated, compact groups (see Fig. 4.1). For elongated clusters, the minimum variance method would perhaps cut the clusters in two in its search for compactness (Wishart, 1969, shows such an example using astronomical data). The single link method - with its chaining effect - or the MST would be ideally suited instead. In the case of linked, globular clusters an estimate may be made of the density in the region of each point (e.g. the number of other points within a specified radius: this approach will be taken in mode analysis, below). This will indicate which points form part of the interconnecting links, constitute noise points, or are otherwise to be ignored. For touching globular groups, the

Well-separated, compact groups.

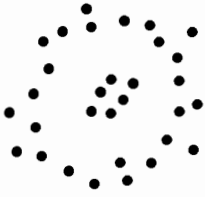


Elongated clusters.

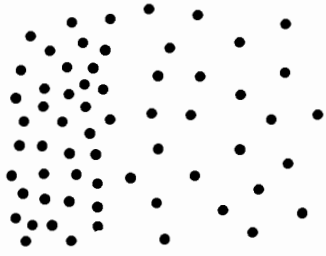
Linked globular groups.



Touching globular groups.



Concentric groups.



Groups characterised by differing densities.

Fig. 4.1 - Point patterns whose constituent groups are to be automatically recognized.

MST may not be of direct use. The minimum variance method should, however, find the clusters, and mode analysis (see below) may also be profitably applied here. Finally, in the cases of concentric groups, or of groups characterised by differing densities, the MST may be used. In the former case, a large link will indicate where the MST ought to be broken in order to leave the two components. In the latter case, a histogram of edge dissimilarity weights should uncover two distinguishable sets of dissimilarities: edges of small dissimilarity will relate to the high density part of the point pattern, and edges of greater dissimilarity will relate to the low-density region. Deleting all edges of dissimilarity weight greater than some thresold in the MST causes the resultant tree to connect only high-density points. Similarly, the low-density component may be ascertained. Note, though, that some extra treatment may be required for points on the boundary of the two regions in order to avoid misclassification (see Zahn, 1971).

Recent research has branched into two directions. On the one hand, a "shortest spanning path" (i.e. a path, spanning all points, which is as short as possible in totalled dissimilarities) achieves many of the same results as does the MST, but with greater computational ease: see Slagle et al. (1974), Slagle et al. (1975) and Lee (1981). On the other hand, other graph theoretical structures have been used with which the MST may be related as a special case: see Jarvis and Patrick (1973), Urquhart (1982), Sibson (1980), Ahuja (1982) and Fairfield (1983). Although these approaches are of importance and offer advantages in specific problems, the MST remains of great interest as a general-purpose technique.

4.3 TRADITIONAL MINIMAL SPANNING TREE ALGORITHMS

The algorithm for the single linkage hierarchical clustering described in Chapter 1 required $O(n^2)$ storage space for the dissimilarity matrix, and $O(n^2)$ processing - $O(n)$ iterations, each necessitating $O(n)$ updating of the dissimilarity matrix. These performance results may be considered as baseline results. They appear to be very satisfactory since the input string presented to the algorithm is $O(n^2)$ long. However they may be bettered, without detriment to the exactness of the output, using algorithms described in sections 4.4 and 4.5.

The algorithms described in this section may be regard as alternatives to that described in Chapter 1; and they construct a MST rather than directly building the single link hierarchy. Their performance results do not improve on the performance of the algorithm looked at in Chapter 1. However they are of particular importance for two reasons. Firstly, the computationally efficient algorithms described in section 4.4 (probably the most efficient, general-purpose, current algorithms) are directly inspired from these algorithms. Secondly, it may be necessary to construct a MST or a single link hierarchy, on a sparse graph. This is a graph where the number of edges, m, is less than $n(n-1)/2$ and so a performance result is desired in terms of m and n. In Chapter 5, application-areas for such a problem will be described. Again, for this problem, efficient algorithms which are described in section 4.5 of this chapter are direct derivations of the algorithms now described.

In section 4.4 a single fragment algorithm allowing for the incorporation of fast NN-finding techniques will be studied. The Prim-Dijkstra algorithm, by comparison, involves brute-force NN-searching. It is assumed that the algorithm will work on the stored matrix of dissimilarities, requiring $O(n^2)$ space.

Algorithm A. Prim-Dijkstra MST algorithm

Step 1. Find the closest vertex to an arbitrary vertex. Call these two vertices a fragment of the MST.

Step 2. Determine the closest vertex, not in the fragment to any vertex in the fragment.
Add this vertex to the fragment.

Step 3. If all n vertices are not included in the fragment then return to Step 2.

There are $O(n)$ iterations (Steps 2,3) in this algorithm since a MST must contain n-1 edges. Ordinarly Step 2 will require $O(n^2)$ operations, leading to an overall complexity of $O(n^3)$. An $O(n^2)$ implementation may be achieved as follows. For each vertex not in the fragment, maintain the closest vertex to it which is in the fragment. Initially $O(n^2)$ operations are required for this. On each iteration (Step 2), as a vertex (say, v) is added to the fragment, check to see if any nearest neighbour (among fragment members) of a vertex outside the fragment can now be bettered by v. This requires checking $O(n)$ values, but allows the implementation of Step 2 in $O(n)$ time.

The foregoing algorithm grew a single fragment through n-1 iterations. As an alternative approach, it is often fruitful to attempt to build the desired structure in a parallel fashion, - by allowing more than one fragment which will be subsets of the eventual MST. Sollin's algorithm does this, again assuming prior calculation of all dissimilarities (edge weights) which requires $O(n^2)$ or $O(m)$ operations. (In section 4.4 a multiple fragment algorithm bypassing this requirement will be described). An example of this algorithm is to be seen in Fig. 4.2.

Consider all vertices, to begin with, as singleton fragments of the MST.

Algorithm B. Sollin's MST algorithm.

Step 1. For each fragment in turn, determine its closest fragment. All edges so specified will be part of the MST.

Step 2. Scan each of these edges, deleting the corresponding fragments from the list of fragments, and placing the new merged fragment onto the fragment-list.

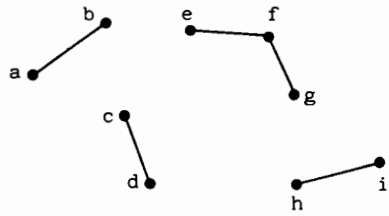
Step 3. While the fragment-list has more than one member, return to Step 1.

The closeness relation in Step 1 is not necessarily symmetric: in fact we have here a generalization of NN-graphs (see Chapter 3), where a RNN pair (symmetric) or part of a NN-chain (asymmetric) may be found among fragments. These least cost edges between fragments, determined in Step 1, must be part of the MST. This may be shown as follows. By construction we have minimal cost, but do we have a tree? If a cycle were possible we would have, for example, f_2 (fragment 2) as NN to f_1, f_3 as NN to f_2, and f_1 as NN to f_3. But then the least interconnecting link between f_2 and f_1 is greater than the least interconnecting link between f_3 and f_2 (otherwise f_1 would be NN to f_2), which in turn is greater than the analogous link between f_3 and f_1. Here we have a contradiction since by definition the link between f_1 and f_2 is less than that between f_3 and f_1. This proof may be extended to cycles with more than 3 vertices. Note that care must be taken in programming this algorithm to incorporate arbitrary choice-making in the case of equal dissimilarities, in order to avoid the creation of cycles.

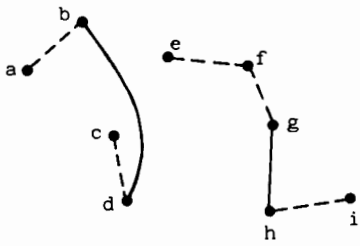
In Step 1 there are $O(n)$ operations for each of n vertices. At most $\lceil n/2 \rceil$ (the least integer $\geqslant n/2$) fragments are produced. On the next pass through Step 1, $O(n^2)$ operations will be again required, leading to at most $\lceil n/4 \rceil$ fragments. Continuing, it is seen that there are $O(\log n)$ iterations yielding $O(n^2 \log n)$ performance.

	a	b	c	d	e	f	g	h	i
a	0	1	5	6	6	5	7	5	6
b	1	0	5	3	6	8	5	7	5
c	5	5	0	2	5	6	8	6	6
d	6	3	2	0	5	6	4	7	5
e	6	6	5	5	0	2	3	6	7
f	5	8	6	6	2	0	1	5	5
g	7	5	8	4	3	1	0	3	6
h	5	7	6	7	6	5	3	0	1
i	6	5	6	5	7	5	6	1	0

Given matrix of dissimilarities between 9 vertices.



Using the lightest edges incident on each vertex, the above four fragments are found. The set of lightest edges from these fragments yield:



The final lightest edge between fragments connects d to g yielding MST.

Fig. 4.2 - Sollin's algorithm for constructing a MST.

In the case of a sparse graph, incident edges may be stored as linked lists, or as some other convenient data structure. Step 1 will require a scan of all edges on each occasion, i.e. $O(m)$ operations. Thus the performance in this case is $O(m \log n)$.

4.4 MINIMAL SPANNING TREE USING FAST NEAREST NEIGHBOUR SEARCHING

In Chapter 3, single and multiple cluster algorithms for geometric clustering methods have been described, which allowed for the incorporation of fast NN searching routines. In this section, suggested approaches to single link hierarchical clustering will be discussed. These approaches are based on the MST (which may be subsequently transformed into the hierarchy).

The following proposition has been used in Chapter 3:

Proposition 1: Given a point-set, any pair of RNNs is a class or cluster of an agglomerative hierarchic clustering, if the hierarchic clustering method satisfies the reducibility property.

The single link method satisfies this property, but a stronger result is:

Proposition 2: Any NN-chain is a subset of MST.

Any NN-chain, originating in an arbitrary point and ending in a pair of RNNs, therefore defines in reverse order a sequence of nested clusters in a single linkage hierarchic clustering. Instead of single and multiple cluster algorithms for geometric cluster methods, we have here single and multiple fragment algorithms, where a fragment is "grown" from a NN-chain. The following is a single fragment algorithm.

Algorithm C. MST algorithm using NN-chains.

Step 1. Construct a NN-chain; let q be the last point added, and call the NN-chain a fragment.

Step 2. Find r, the nearest point to q which is not in the fragment.

Step 3. If, for some i in the fragment, $d(i,NN(i)) < d(q,r)$ then see if there is an s not in the fragment such that

$$d(i,NN(i)) < d(i,s) < d(q,r).$$

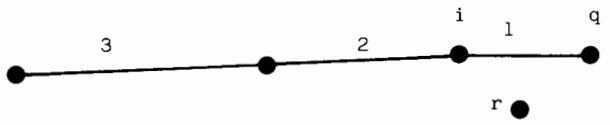
If so, find the least such d(i,s), and connect s to i; otherwise connect r to q.

Step 4. Redefine q to be the point whose link to the fragment is of least dissimilarity, and return to Step 2 until all points are in the fragment.

Step 3 is explained as follows (cf. Fig. 4.3). The point r could be connected to q. However it must be checked if a closer point could instead be connected to some other point in the fragment. The edge (i,NN(i)) is in the fragment if i is in the fragment. If $d(i,NN(i)) > d(q,r)$, and if some s is connected to i then $d(i,s) < d(q,r)$, which together imply: $d(i,s) < d(i,NN(i))$. But then s is the NN of i, and from this contradiction it is seen that we were justified in connecting r to q.

Step 3 possibly necessitates "climbing back" some way in the fragment (i.e. it requires the finding of NNs of a number of vertices in the fragment) and is best implemented using a list of vertices in the fragment, ordered by the smallest dissimilarity which connects them to the fragment. The analysis of this algorithm depends on the average number of iterations between Steps 2 and 3, - i.e. if constant or O(n), we get overall complexity of $O(n^2)$ or $O(n^3)$, respectively. An approximate minimal spanning tree algorithm, where this number of iterations is held constant, may be adequate but in general worst-case O(n) must be assumed.

A better, multiple fragment algorithm with two separate stages (Steps 1, 2, and 3; and Steps 4 and 5) is as follows.



Final two nodes of fragment are i and q.

NN(i) = q.

d(i,q) = 1.

Closest node to q, not in fragment, is r.

d(q,r) = 1.9.

Search for closest node to i, not in fragment.

Find r: d(i,r) = 1.8.

Therefore s (see description of Algorithm C) is q, and q is connected to i.

Fig. 4.3 - Example of Algorithm C.

Algorithm D. Parallel MST algorithm.

Step 1. Pick an arbitrary point.

Step 2. Construct a NN-chain from this point.

Step 3. Pick another isolated point, and return to Step 2 until all points are in one of p NN-chains.

Step 4. Connect the closest point in an arbitrary NN-chain (or fragment) to some other NN-chain (or fragment), using Steps 2, 3 and 4 of Algorithm C.

Step 5. Return to Step 4 until all points are in one fragment.

Algorithm D will work better if the NN-chains are long, i.e. if the points chosen in Steps 1 and 3 are in sparse regions. Following the iterated Steps 2 and 3, there are p NN-chains and hence p-1 edges remaining to be found in the minimal spanning tree.

The principal computational advantage of Algorithms C and D lies in their ability to incorporate efficient NN-finding techniques. The latter algorithm has been found to be of O(n log n) average complexity, when a MDBST approach (see Chapter 2) is employed (Bentley and Friedman, 1978).

We will conclude this section with an adaptation of Algorithm C which is suitable for any graph (or linkage) method (e.g. the complete or average linkage methods described in Table 1, section 3.3). For these methods, inter-cluster dissimilarity cannot be calculated in O(1) time unless we have the entire set of dissimilarities directly accessable. Algorithm C may be amended as follows.

Algorithm E. Algorithm for any linkage-based method.

Step 1. Select an object arbitrarily.

Step 2. Determine and store all inter-object dissimilarities.

Step 3. Grow the NN-chain from the object chosen, until a pair of RNNs are obtained.

Step 4. Agglomerate these objects.

Step 5. Update the dissimilarity table, using the Lance-Williams formula.

Step 6. From the node in the NN-chain which preceded the RNNs, or from an arbitrary node (object or cluster) if the NN-chain is empty, go to Step 3 until only one node remains.

Clearly, $O(n^2)$ storage is required here. As before, there are $O(n)$ growths of the NN-chain, each requiring $O(n)$ operations; plus $O(n)$ contractions of the NN-chain, each requiring $O(n)$ updating of the dissimilarity table. In total, therefore, computational complexity is $O(n^2)$.

4.5 MINIMAL SPANNING TREE OF SPARSE AND PLANAR GRAPHS

Special cases of the MST problem arise when not all edges exist. In a sparse graph, the number of edges may not be $O(n^2)$. The first algorithm discussed below has computational complexity O(m log n). Thus this result comes close to O(m), and m might be quite small. The number of edges is small in the case of planar graphs: the second algorithm discussed in this section achieves the very satisfactory computational performance of O(n). These algorithms are due to Cheriton and Tarjan (1976).

Let us begin with the problem of sparse graphs where we seek the best performance in terms of n and of m (bearing in mind that the latter will always be greater for non-degenerate problems). The following algorithm is a particular implementation of Sollin's algorithm (see section 4.3, Algorithm B).

For each vertex v in the vertex-set V, let the number of incident edges be denoted by n_v. Thus, $\sum_{v \in V} n_v = 2m$

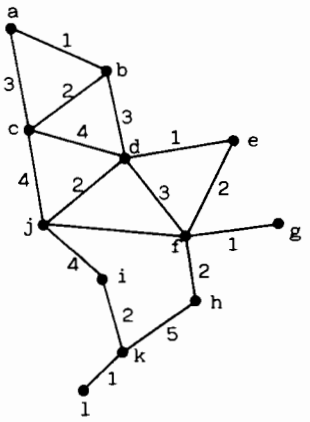
A preprocessing stage to Algorithm B is as follows. Divide each set of n_v edges into groups of size k ($\leq$ k for the final group: for simplicity, we will assume that every group contains precisely k edges). Sort each of these groups. For each vertex, we have n_v/k groups, and the sort operation will require O(k log k) comparisons. Hence (n_v/k) k log k will be the order of magnitude of the number of operations required for vertex v. For all vertices, the number of operations required is of the order of $\sum_{v \in V} n_v \log k = 2m \log k$.

Algorithm B is now implemented as follows. Determining the lightest edge

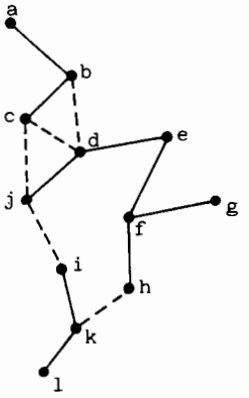
incident on a vertex requires $O(n_v/k)$ comparisons since this edge is to be found in some one of the n_v/k sorted groups. The lightest edges incident on all vertices are therefore obtained in $O(m/k)$ operations. When two vertices (later: fragments) are merged, their associated sorted groups of incident edges are simply appended together so that all such groups remain internally sorted. On subsequent executions of Step 1 in Algorithm B, again $O(m/k)$ processing is required. (Note that we may discard from all sorted groups those edges connecting vertices in the same fragment: in the entire algorithm, this cannot surpass the deleting of 2 m edges). Thus, overall, Algorithm B's complexity becomes $O(m/k \log n) + O(m \log k)$ where the latter term, as was seen above, is required for preprocessing. By chosing $k = \log n$, the second of these terms dominates and gives complexity $O(m \log \log n)$ for this implementation.

We turn attention now to constructing a MST from a planar graph. The following graph-theoretic result will be of central importance in the analysis of an algorithm for building the MST of a planar graph: for a planar graph, $m = O(n)$. Specifically, $m \leq 3n-6$ for $m > 1$. For proof, see for example Tucker (1980).

Referring to Sollin's algorithm (Algorithm B of section 4.3), $O(n)$ operations are required to establish the least cost edge from each vertex (since there are only $O(n)$ edges present). On the subsequent execution of Step 1, we may define a new, planar graph with as new vertices the fragments found so far (see Fig. 4.4). There will be at most $\lceil n/2 \rceil$ such new vertices. $O(n/2)$ processing will serve to replace multiple edges between the same pair of (new) vertices with the minimum edge weights, since the total number of multiple edges remains $O(n)$. Following this, $O(n/2)$ processing is required to merge fragments (Step 1 of Algorithm B). Continuing, we obtain the overall complexity as being $O(n)+O(n/2)+O(n/4)+ + \ldots = O(n)$.



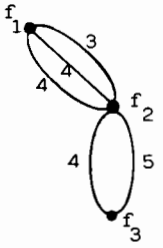
Given planar graph.



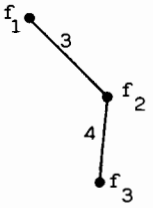
Heavy lines show fragments found in first iteration of algorithm.

Clean up of deletable edges has been carried out.

Dashed lines are edges connecting fragments.



New graph consisting of three fragments.



New graph with multiple edges removed.

Fig. 4.4 - Stages in an implementation of Sollin's algorithm on a planar graph.

4.6 EXTENSION: MODE ANALYSIS

The MST (and the single link method) use dissimilarities or distances. One difficulty with distance-based procedures is that any regularity in the data may give rise to many identical distances and to subsequent degenerate or misleading cluster results. This difficulty can be bypassed by incorporation of local (e.g. density) information, which will indicate which points are of greater importance. In general, we may use

- node weights: valuations on the points under consideration; or
- node and edge weights, where interpoint dissimilarities are additionally used.

We will begin with algorithms in the first category.

Node weights used in pattern recognition have generally involved an estimate of density at each point. Among such node weights are the following:

1) $|N(i)|$ where $N(i) = \{ j \mid d_{ij} \leqslant r \}$

N(i) is the neighbourhood of point i, defined here as the set of points within radius r of i. The weight of node i is the cardinality of its neighbourhood.

2) $1/k \quad \Sigma \{ d_{ij} \mid j \in N(i) \}$

where N(i) is the set of k nearest neighbours of i. The weight of node i, here, is the average distance to the k-nearest neighbours; it is a measure of potential, i.e. the inverse of density. Unlike the previous node weight, it is independent of the scaling of the original data.

3) In image processing nodes corresponding to pixels may be weighted by

the grey level intensity at that point. This is the most immediate node weighting scheme. Others may be specified, though, such as a measure of edge gradient at that pixel. The edge value is the difference in intensities between contiguous pixels; and edge gradient is the maximum such value between a pixel and its neighbours.
The edge gradient may be useful for contour extraction, i.e. for determining significant boundaries.

The use of dissimilarity d, in the above, is almost invariably Euclidean, - the most natural choice for visual patterns of points. Let f_i be the weight associated with node i, using some one of the above definitions.

The most straightforward approach to the clustering of node-valued graphs is to use a single threshold: nodes of density-weight greater (or potential-weight less) than the threshold are members of the same cluster, if they are in addition contiguous to at least one other member of the cluster. By decreasing the threshold in the case of densities, or by increasing it in the case of potential, a hierarchy of embedded classes is obtained.

The clustering brought about by thresholding can also be expressed in terms of more traditional distance-based clustering. Define $\delta_{ij} = \infty$ if i and j are not contiguous; otherwise define $\delta_{ij} = - \min \{f_i, f_j\}$ where f is a density. As values of f are examined in increasing order of magnitude, i will be connected to j only if both f_i and f_j are greater than the density threshold. This clustering method may therefore be viewed as a single link method. Clustering by thresholding in the manner described is a common technique in image processing; and it has also been used for wealth data for geographic regions (i.e. f_i = per capita income for region i; see Hartigan, 1975).

The use of node weights (such as point densities, attributes of populations or states, etc.) is an intuitively clear starting point from which to carry out the automatic grouping process. But the use of inter-point distances, while being fraught with difficulty when many distances are identical, nonetheless allows a more fine-tuned analysis: for example, in the threshold-based clustering described above, no account is taken as to whether a new addition to a cluster is closely related to one or to many of the cluster members. In order to allow for varying degrees of relationship, a dissimilarity may be recreated from the node weights. One possibility for this is to construct directed arcs defined by $\delta_{ij} = f_j - f_i$ where i and j are contiguous. Therefore if $f_j > f_i$ then δ_{ij} is directed from i to j, while if $f_j < f_i$ then the arc is negatively weighted, and so is directed from j to i. Rather than the difference in densities, as this dissimilarity coefficient is, the density gradient (difference in density per unit distance) has usually been preferred (see e.g. Koontz et al., 1976). This is given by $\delta_{ij} = (f_i - f_j)/d_{ij}$ where d is the Euclidean distance, and δ is again an asymmetric dissimilarity. A generalization of the single linkage method (or the MST) has been used for such dissimilarities. It is to connect i to j if δ_{ij} is positive and maximum among nodes j which are contiguous to i, i.e. to construct components such that the density gradient is always upwards.

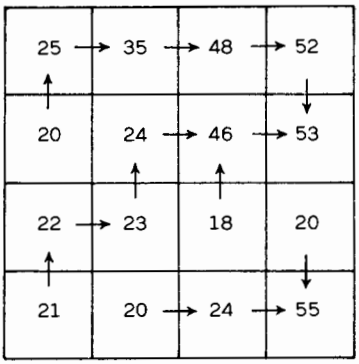
It is easily verified that each such component is a directed tree, so long as no δ_{ij} equals zero. In order to facilitate subsequent labelling and other processing of the components, cycles in the directed graph must be prevented, and arbitrarily directed edges are formed for $\delta_{ij} = 0$ following a test that a cycle will not result. Note that in this approach each component nominates a unique "centre" or local peak in density. It has also been proposed that local valleys in density are equally revealing of structure in the data (see Johnston et al., 1979). Such an alternative

viewpoint on the data may be carried out by simply defining δ_{ij} as the negative of the mode-oriented approach.

A similar approach - determining components which are directed trees - has been used in image processing (see Fig. 4.5). Narendra and Goldberg (1980) define as a weight at each pixel (node) a measure of edge gradient. Having thus a value for f_i, the asymmetric dissimilarity δ_{ij} is constructed and the directed tree formed in the manner described above.

Another very different application of this directed forest approach has also been successfully employed, as follows. A histogram of intensities often permits visually different parts of the image to be distinguished, - different modes in the histogram correspond to distinct, but significantly numerous, sets of pixel intensities. The gradient climbing procedure, used in point pattern recognition, also allows the modes of the histogram to be determined (see Fig. 4.6). Smoothing of the histogram might be required - using for instance a 3-point moving average - and Wharton (1983) suggests an "adaptive smoothing" where non-mode parts of the histogram (below a user-specified threshold value) alone are smoothed in this way. For 4-band LANDSAT data, a 4-dimensional generalization of this approach has been employed by constructing a 4-dimensional histogram. This is simply a grid of regular cells in 4-dimensional space, each containing the frequency of occurrence of associated 4-valued pixel intensity vectors.

Note that the dissimilarity constructed in the foregoing examples has been anti-symmetric (i.e. $\delta_{ij} = -\delta_{ji}$). Other asymmetric (but not anti-symmetric) coefficients may also be constructed for point pattern recognition. For instance, Ozawa (1983) defines $\delta_{ij} = f_i \cdot \exp(-b \cdot d_{ij})$ where b is some scale constant. This dissimilarity will yield different values



Directed trees

1	1	1	1
1	1	1	1
1	1	1	2
1	2	2	2

Class labels

Fig. 4.5 - Clustering by connecting pixels to highest-valued pixel among four-neighbours.

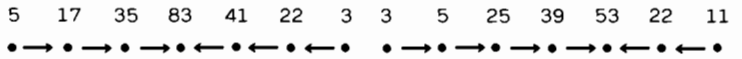


Fig. 4.6 - Clustering by gradient climbing to distinguish modes of a histogram (given here by the set of frequencies).

for δ_{ij} and for δ_{ji} depending on density defined at i and at j. Katz and Rohlf (1973) use another asymmetric coefficient.

This brief look at mode analysis indicates the range of approaches in an area which continues to expand rapidly. Many of the approaches described required local processing to establish the node or edge weights, and the connectivity clustering was then performed with reference only to the neighbourhood of the point (or pixel or other object). Thus these algorithms would appear to be well-suited to parallel implementations. A problem to be solved in certain cases concerns the improvisation required to arbitrarily choose among equally-valued dissimilarities.

4.7 REFERENCES

N.AHUJA, Dot pattern processing using Voronoi neighbourhoods. IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-4, 336-343 (1982).

J.L. BENTLEY and J.H. FRIEDMAN, Fast algorithms for constructing minimal spanning trees in coordinate spaces. IEEE Transactions on Computers C-27, 97-105 (1978).

D. CHERITON and R.E. TARJAN, Finding minimum spanning trees. SIAM Journal of Computing 5, 724-742 (1976).

J. FAIRFIELD, Segmenting dot patterns by Voronoi diagram concavity. IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-5, 104-110 (1983).

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

N. JARDINE and R. SIBSON, Mathematical Taxonomy, Wiley, New York (1971).

R.A. JARVIS and E.A. PATRICK, Clustering using a similarity measure based on shared near neighbours. IEEE Transactions on Computers C-22, 1025-1034 (1973).

B. JOHNSTON, T. BAILEY and R. DUBES, A variation on a nonparametric clustering method. IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-1, 400-408 (1979).

J.O. KATZ and F.J. ROHLF, Function-point cluster analysis. Systematic Zoology 22, 295-301 (1973).

W.L.G. KOONTZ, P.M. NARENDRA and K. FUKUNAGA, A graph-theoretic approach to nonparametric cluster analysis. IEEE Transactions on Computers C-25, 936-944 (1976).

R.C.T. LEE, Clustering analysis and its applications. In Advances in Information Systems Science, Edited by J.T. Tou, Vol. 8, pp. 169-292, Plenum Press, New York (1981).

P.M. NARENDRA and M. GOLDBERG, Image segmentation with directed trees. IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-2, 185-191 (1980).

K. OZAWA, CLASSIC: a hierarchical clustering algorithm based on asymmetric similarities. Pattern recognition 16, 201-211 (1983).

F.J. ROHLF, Algorithm 76: hierarchical clustering using the minimum spanning tree. The Computer Journal 16, 93-95 (1975).

F.J. ROHLF, Single link clustering algorithms. In P.R. Krishnaiah and L. N. Kanal (Editors),Handbook of Statistics Vol.2, pp. 267-284, North-Holland, Amsterdam (1982).

R. SIBSON, The Dirichlet tesselation as an aid in data analysis. Scandinavian Journal of Statistics 7, 14-20 (1980).

J.R. SLAGLE, C.L. CHANG and R.C.T. LEE, Experiments with some cluster analysis algorithms. Pattern Recognition 6, 181-187 (1974).

J.R. SLAGLE, C.L. CHANG and S.R. HELLER, A clustering and data-reorganizing algorithm. IEEE Transactions on Systems, Man, and Cybernetics SMC-15, 125-128 (1975).

P.H.A. SNEATH and R.R. SOKAL, Numerical Taxonomy. Freeman, San Francisco, 1973.

A. TUCKER, Applied Combinatorics. Wiley, New York, 1980.

R. URQUHART, Graph theoretical clustering based on limited neighbourhood sets. Pattern Recognition 15, 173-187 (1982); Erratum, Pattern Recognition 15, 427 (1982).

D. WISHART, Mode Analysis: a generalization of nearest neighbour which reduces chaining effects. In Numerical Taxonomy, Edited by A.J. Cole, Academic Press, London, pp. 272-281 (1969).

S.W. WHARTON, A generalized histogram clustering scheme for multidimensional image data. Pattern Recognition 16, 193-199 (1983).

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

CHAPTER 5 - NEW CLUSTERING PROBLEMS

5.1 INTRODUCTION

Classification is such an all-embracing human activity that practically no area of automated data processing can dispense with some form of clustering. Two relatively new areas, which use algorithms adapted from those explored in previous chapters, are studied in sections 5.2 and 5.3.

A clear example of the contiguity-constrained clustering problem (section 5.2) is the grouping of people/areas on the basis of some given set of socio-economic attributes. It might be expected that the objects of analysis which come from major urban areas would be grouped together. Consider now the presence of a contiguity-constraint: the resulting clustering ought to clearly demarcate the urban areas, and instead group with them their respective hinterlands. A good background study in this area is Fischer (1980). Gordon (1980) should be consulted for another perspective on the problem of contiguity constraints in clustering.

The problem of clustering interaction data (section 5.3) is that of handling asymmetric proximities. Slater (1981) or Masser and Scheurwater (1980) provide illustrative studies.

5.2 CONTIGUITY-CONSTRAINED CLUSTERING

One major theme in clustering research over the past two decades has been the automatic classification of quantitatively described objects, without any constraint as to which pairs of such objects might ultimately find themselves in the same class. A second recent trend in clustering work has been where there is such an inherent or an imposed representational constraint. In this section we review general-purpose algorithms which have the function of segmenting (or regionalizing, or zoning) a set of objects, each described by a descriptor vector.

Contiguity-constrained clustering uses proximities between objects, defined in descriptor space, and also takes into account contiguous neighbourhoods. Depending on the application, the contiguous neighbourhood is defined in different ways. In image processing, where the image consists of pixels characterized by grey-level intensity values, the 8 neighbouring pixels (east, north-east, north, etc.) are suitable candidates. Similarly with agricultural data, the terrain which is characterised by crop yields or chemical constituents may be subdivided into square parcels and the neighbourhood of a parcel may be defined as its 8 adjacent parcels. With point patterns, a radius may be used to define the neighbourhood of point i: $N(i) = \{j \mid d_{ij} \leqslant r\}$, and j is said to be contiguous to i; a neighbourhood may alternatively be defined as the k nearest neighbours of an object (see section 4.6 of Chapter 4). In general, when the objects do not comprise the squares of a regular grid, it is convenient to express the contiguity relationship as a binary matrix, with a contiguity value $c_{ij} \in \{0,1\}$ defined on all pairs of objects. Such a matrix can be externally defined by the user, - for example, in the case of contiguities between bordering countries (characterised, perhaps, by socio-economic attributes) or other basic spatial units.

If the stepwise agglomerations in hierarchical clustering are constrained to be between clusters which are contiguous, the problem of inversions (reversals or non-monotonic increase/decrease in cluster criterion value) is likely. This is when $d(q \cup r, s) \not\geq d(q,r)$ for three clusters q,r, and s, where q and r agglomerate to form $q \cup r$, and where the cluster criterion value (e.g. compactness or connectivity) is related to the dissimilarity d between clusters (cf. section 3.2 of Chapter 3). In using the common clustering criteria (e.g. as listed in Table 1, section 3.3), with the restriction that only contiguous clusters can merge, inversions tend to arise since a previously forbidden merger between two very similar classes may be permitted by changes in the contiguity relation. The presence of inversions in a hierarchy is disadvantageous: it makes difficult the interpretation of partitions, and the definition of dissimilarity between classes. Only two of the traditional hierarchical clustering methods appear to be amendable in order to permit agglomerations between contiguous clusters, and simultaneously guarantee that no inversions will arise. These methods are the single and complete linkage methods, which will use two different approaches to the updating of the contiguity relation following each agglomeration.

The contiguity-constrained single linkage method is as follows: at each agglomeration, fuse together the two clusters of least interconnecting dissimilarity, such that this dissimilarity is between a pair of contiguous objects. Initially all clusters are singletons. Each agglomeration in this method is necessarily between a pair of contiguous objects. Therefore, given the contiguity graph where each edge connecting a pair of contiguous objects is weighted by the dissimilarity (in descriptor space) between the objects, it is seen that the minimal spanning tree of the weighted contiguity graph may be obtained and subsequently transformed into the single linkage hierarchy. A simple proof that the

contiguity-constrained single linkage hierarchy cannot present inversions is to replace the dissimilarities between all pairs of non-contiguous objects by some arbitrarily large value. The construction of the single linkage hierarchy on this amended set of dissimilarities is well-defined (in the sense that at all stages the traditional algorithm can be employed and, assuming the contiguity graph is connected, infinite dissimilarities will never be used as cluster criterion - i.e. connectivity-values). As in the case of the usual single linkage method, it has been found that this method has a pronounced tendency to "chain", i.e. to successively agglomerate singletons to one, large cluster in each partition (see Fischer, 1980). Efficient algorithms for constructing a constrained single linkage hierarchy have been examined in section 4.5 of Chapter 4.

An alternative approach for contiguity-based agglomerative clustering allows agglomeration of any pair of clusters such that there exists a contiguity link between at least one member of each of the clusters. Such a definition of contiguity has generally been used in incorporating a contiguity constraint in the minimum variance (Ward's) method. However no way of using this method, in an inversion-free manner, has yet been found. For a review of work in this direction, see Murtagh (1984). Of the major hierarchical methods, only the complete link method excludes the possibility of inversions when constrained in this manner. Before showing this, it may be remarked that the $O(n^2)$ time and $O(n^2)$ space algorithm of section 4.4 (algorithm E) is easily updated to include an additional testing of contiguity whenever a linkage in the NN-chain is created.

Proposition: The complete link method, with the constraint that at least one member of each of the two clusters to be agglomerated be contiguous, is guaranteed not to give rise to inversions.

The proof of this proposition will conclude this section.

Consider three clusters (possibly singletons), q, r, and s. At some stage of the agglomerative construction of the hierarchy, q and r cluster (supposition I); and later, s clusters with q U r directly and not through the intermediary of some cluster, t (supposition II). It is seen that no loss of generality ensues with supposition II, since inversion-free agglomeration of q U r and s implies inversion-free agglomeration of q U r and t, followed by inversion-free agglomeration of q U r U t and s. Also, without loss of generality, assume that $d(q,s) \leqslant d(r,s)$. Three cases may now be considered.

Case I : $d(q,s) \leqslant d(r,s) \leqslant d(q,r)$.
If either (q,s) or (r,s) are contiguous, they should have clustered prior to (q,r): this is contrary to supposition I. If neither (q,s) nor (r,s) are contiguous, then s cannot cluster with q U r, except through both s and q U r being contiguous with some other cluster, t: this is contrary to supposition II.

Case II: $d(q,r) \leqslant d(q,s) \leqslant d(r,s)$.
(q,r) must be contiguous so that they may cluster as supposed. If neither (q,s) nor (r,s) are contiguous, supposition II is not possible. If either are contiguous, then s can cluster with q U r without giving rise to an inversion.

Case III: $d(q,s) \leqslant d(q,r) \leqslant d(r,s)$.
For q and r to cluster before q and s, it must be assumed that q and s are not contigous; supposition I implies that (q,r) is contiguous; and supposition II implies that (r,s) is contiguous.

We may summarize: case I cannot arise; case II presents no problem; and case III is the only case to possibly give rise to an inversion. An

inversion arises in case III if $d(q,r) \not\leq d(q \cup r, s)$. Traditional clustering strategies (cf. Table 1 of section 3.3) define $d(q \cup r, s)$ as a function of $d(q,r)$, $d(q,s)$ and $d(r,s)$. Therefore, choosing $d(q \cup r, s)$ as $\max \{ d(q,s), d(r,s) \}$ precludes an inversion in case III.

5.3 CLUSTERING OF INTERACTION DATA

Interaction flow matrices are square, asymmetric matrices which arise in many of the social sciences. Examples of the flows or interactions involved in such tables are: industrial inputs and outputs for a set of firms or countries; cross-citations for a set of journal articles; internal migrations or trips for a set of geographic regions; or occupational mobility data for a set of occupations of a given population for a set time-period. In some of these applications contiguous clusters may be required, especially in the case of data with geographic location information. Two types of clustering problem may be considered. Consider the case of journey-to-work data, with a given set of zones and associated numbers of cross-boundary journeys. We may wish to ascertain nodal or "central" zones (i.e. those that receive large numbers of workers), or alternatively to carry out a regionalization of the given zones into a smaller set of homogeneous areas. For these two different problems, two approaches have been suggested. A variant of the single linkage method has been proposed for the former problem, - the determining of nodal zones. Faithful representation of the asymmetric character of the interaction matrix is the primary objective, and one disadvantage of this approach is the "chaining" side-effect of single linkage clustering. For the second problem, - creating homogeneous zones - variants of the average linkage method have been used. A disadvantage here is the conflict between the clusters of zones and the often asymmetric characteristics of these zones (i.e. in-flows greater than out-flows or viceversa).

In both cases a standardization of the given flows is carried out, in order to adjust for disproportionate flow in large zones. In the case of the compact clustering, this has been achieved by dividing every element of the flow array by the corresponding row and column sums

(see below). In the case of directed single linkage, analogous frequencies have been obtained. Since the initial, asymmetric data may be standardized directly, an iterated (re) calculation of row (column) sums is carried out until the row (and column) sums are all identical.

The directed linkage procedure involves a generalization of the single linkage method for dealing with asymmetric proximities (here: the standardized flows). The strong components of the directed graph are the sets of mutually reachable nodes (or zones): each node can be reached from another in the same component if there is a series of consistently directed arcs from one to the other. As in the case of the single linkage method, a dendrogram may be constructed, corresponding to the components formed at differing thresholds of proximity. The example shown in Fig. 5.1 is from Tarjan (1983). Note that following each agglomeration, all directed edges from vertices of the new cluster to an outsider vertex may be replaced by the least-weighted edge among them; this, together with a similar updating of in-flows to the new cluster, allows the cluster's vertices to be replaced by a single vertex. Hence, updating after each of the (at most) n-1 agglomerations requires O(n) time. In order to find which pair of vertices to merge, a sorted list of edges may be used (requiring O(m log n) time: i.e. sorting m values, where $m \leqslant n(n-1)/2$). Resulting complexity is then $O(n^2)$ or O(m log n), depending on which term dominates. If m is much less than n, then the O(m log n) algorithm described by Tarjan (1983) should be used.

For constructing a hierarchy of compact clusters, an algorithm proposed by Domengès (1982) is as follows. A symmetric matrix is constructed by summing the $(i,j)^{th}$ elements, for all i and j. Next, the asymmetric matrix is standardized by dividing the $(i,j)^{th}$ element by the product of the associated row and column sums. Finally the sequence of agglomerations takes place by successively seeking the greatest stand-

ardized symmetric flow between regions. Let the symmetric matrix be defined from the given flow matrix by $s_{ij} = f_{ij} + f_{ji}$. When an agglomeration takes place, the (unstandardized) flows to and from the new region, c, equal the sum of flows to and from the sub-clusters a and b : $s_{cc'} = s_{ac'} + s_{bc'}$, for any other region, c'. If s_c and $s_{c'}$ are the totals of rows c and c' (or columns: the matrix has been made symmetric before all agglomerations), then the agglomerations take place on standardized values, s*:

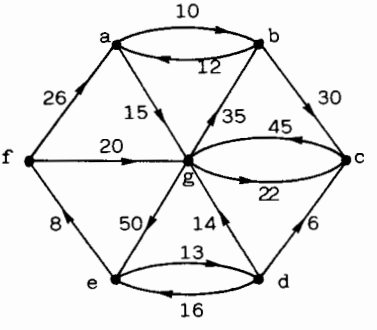
$$s^*_{cc'} = s_{cc'}/s_c s_{c'}$$

$$= (s_{ac'} + s_{bc'})/s_c s_{c'}$$

$$= (s_a s^*_{ac'} + s_b s^*_{bc'})/s_c$$

(simply introducing cancelling terms)

and since $s_c = s_a + s_b$, the above expression ressembles the Lance-Williams update formula for the average linkage (group average or UPGMA) method (cf. Table 1, section 3.3 of Chapter 3).



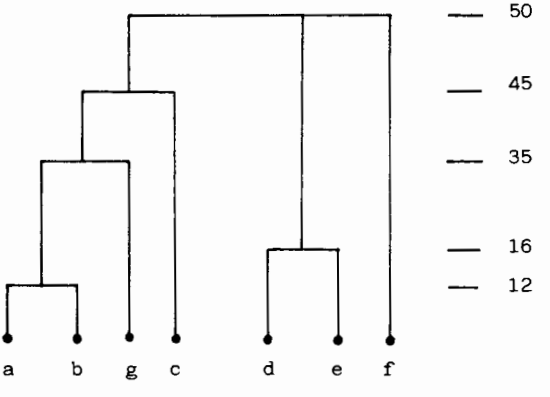


Fig. 5.1 - Example of hierarchical clustering based on strong components at succession of levels.

5.4 REFERENCES

D. DOMENGES, Classification ascendante hiérarchique d'après un critère adapté aux tableaux de flux. Les Cahiers de l'Analyse des Données VII, 169-172 (1982).

M.M. FISCHER, Regional taxonomy. Regional Science and Urban Economics 10, 503-537 (1980).

A.D. GORDON, Methods of constrained classification. In Analyse des Données et Informatique, Edited by R. Tomassone, INRIA, Le Chesnay, pp. 161-171 (1980).

I. MASSER and J. SCHEURWATER, Functional regionalization of spatial interaction data: an evaluation of some suggested strategies. Environment and Planning A 12, 1357-1382 (1980).

F. MURTAGH, A survey of algorithms for contiguity-constrained clustering and related problems. The Computer Journal (in press, 1984).

P.B. SLATER, Combinatorial procedures for structuring internal migration and other transaction flows. Quality and Quantity 15, 179-202 (1981).

R.E. TARJAN, An improved algorithm for hierarchical clustering using strong components. Information Processing Letters 17, 37-41 (1983).