CHAPTER

Clustering Algorithms II: Hierarchical Algorithms 13

13.1 INTRODUCTION

Hierarchical clustering algorithms are of different philosophy from the algorithms described in the previous chapter. Specifically, instead of producing a single clustering, they produce a hierarchy of clusterings. This kind of algorithm is usually found in the social sciences and biological taxonomy (e.g., [El-G 68, Prit 71, Shea 65, McQu 62]). In addition, they have been used in many other fields, including modern biology, medicine, and archaeology (e.g., [Stri 67, Bobe 93, Solo 71, Hods 71]). Applications of the hierarchical algorithms may also be found in computer science and engineering (e.g., [Murt 95, Kank 96]).

Before we describe their basic idea, let us recall that

$$X = \{\boldsymbol{x}_i,\ i = 1, \ldots, N\}$$

is a set of l-dimensional vectors that are to be clustered. Also, recall from Chapter 11 the definition of a clustering

$$\Re = \{C_j,\ j = 1, \ldots, m\}$$

where $C_j \subseteq X$.

A clustering $\Re_1$ containing k clusters is said to be *nested* in the clustering $\Re_2$, which contains $r(<k)$ clusters, if *each* cluster in $\Re_1$ is a subset of a set in $\Re_2$. Note that at least one cluster of $\Re_1$ is a proper subset of $\Re_2$. In this case we write $\Re_1 \sqsubset \Re_2$. For example, the clustering $\Re_1 = \{\{\boldsymbol{x}_1, \boldsymbol{x}_3\}, \{\boldsymbol{x}_4\}, \{\boldsymbol{x}_2, \boldsymbol{x}_5\}\}$ is nested in $\Re_2 = \{\{\boldsymbol{x}_1, \boldsymbol{x}_3, \boldsymbol{x}_4\}, \{\boldsymbol{x}_2, \boldsymbol{x}_5\}\}$. On the other hand, $\Re_1$ is nested neither in $\Re_3 = \{\{\boldsymbol{x}_1, \boldsymbol{x}_4\}, \{\boldsymbol{x}_3\}, \{\boldsymbol{x}_2, \boldsymbol{x}_5\}\}$ nor in $\Re_4 = \{\{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_4\}, \{\boldsymbol{x}_3, \boldsymbol{x}_5\}\}$. It is clear that a clustering is not nested to itself.

Hierarchical clustering algorithms produce a *hierarchy of nested clusterings*. More specifically, these algorithms involve N steps, as many as the number of data vectors. At each step t, a new clustering is obtained based on the clustering produced at the previous step $t-1$. There are two main categories of these algorithms, the *agglomerative* and the *divisive hierarchical algorithms*.

The initial clustering $\Re_0$ for the agglomerative algorithms consists of N clusters, each containing a single element of X. At the first step, the clustering $\Re_1$ is produced. It contains $N-1$ sets, such that $\Re_0 \sqsubset \Re_1$. This procedure continues until the final clustering, $\Re_{N-1}$, is obtained, which contains a single set, that is, the set of data, X. Notice that for the hierarchy of the resulting clusterings, we have

$$\Re_0 \sqsubset \Re_1 \sqsubset \cdots \sqsubset \Re_{N-1}$$

The divisive algorithms follow the inverse path. In this case, the initial clustering $\Re_0$ consists of a single set, X. At the first step the clustering $\Re_1$ is produced. It consists of two sets, such that $\Re_1 \sqsubset \Re_0$. This procedure continues until the final clustering $\Re_{N-1}$ is obtained, which contains N sets, each consisting of a single element of X. In this case we have

$$\Re_{N-1} \sqsubset \Re_{N-2} \sqsubset \ldots, \sqsubset \Re_0$$

The next section is devoted to the agglomerative algorithms; the divisive algorithms are discussed briefly in Section 13.4.

13.2 AGGLOMERATIVE ALGORITHMS

Let $g(C_i, C_j)$ be a function defined for all possible pairs of clusters of X. This function measures the proximity between C_i and C_j. Let t denote the current level of hierarchy. Then, the general agglomerative scheme may be stated as follows:

Generalized Agglomerative Scheme (GAS)

- Initialization:
  - Choose $\Re_0 = \{C_i = \{\boldsymbol{x_i}\},\ i = 1, \ldots, N\}$ as the initial clustering.
  - $t = 0$.
- Repeat:
  - $t = t + 1$
  - Among all possible pairs of clusters (C_r, C_s) in $\Re_{t-1}$ find the one, say (C_i, C_j), such that

$$g(C_i, C_j) = \begin{cases} \min_{r,s} g(C_r, C_s), & \text{if } g \text{ is a dissimilarity function} \\ \max_{r,s} g(C_r, C_s), & \text{if } g \text{ is a similarity function} \end{cases} \tag{13.1}$$

  - Define $C_q = C_i \cup C_j$ and produce the new clustering $\Re_t = (\Re_{t-1} - \{C_i, C_j\}) \cup \{C_q\}$.
- Until all vectors lie in a single cluster.

It is clear that this scheme creates a hierarchy of N clusterings, so that each one is nested in all successive clusterings, that is, $\Re_{t_1} \sqsubset \Re_{t_2}$, for $t_1 < t_2, t_2 = 1, \ldots, N-1$. Alternatively, we can say that *if two vectors come together into a single cluster at level t of the hierarchy, they will remain in the same cluster for all subsequent clusterings*. This is another way of viewing the nesting property.

A disadvantage of the nesting property is that there is no way to recover from a "poor" clustering that may have occurred in an earlier level of the hierarchy (see [Gowe 67]).[1]

At each level t, there are $N-t$ clusters. Thus, in order to determine the pair of clusters that is going to be merged at the $t+1$ level, $\binom{N-t}{2} \equiv \frac{(N-t)(N-t-1)}{2}$ pairs of clusters have to be considered. Thus, the total number of pairs that have to be examined throughout the whole clustering process is

$$\sum_{t=0}^{N-1} \binom{N-t}{2} = \sum_{k=1}^{N} \binom{k}{2} = \frac{(N-1)N(N+1)}{6}$$

that is, the total number of operations required by an agglomerative scheme is proportional to N^3. However, the exact complexity of the algorithm depends on the definition of g.

13.2.1 Definition of Some Useful Quantities

There are two main categories of agglomerative algorithms. Algorithms of the first category are based on matrix theory concepts, while algorithms of the second one are based on graph theory concepts. Before we enter into their discussion, some definitions are required. The *pattern matrix* $D(X)$ is the $N \times l$ matrix, whose ith row is the (transposed) ith vector of X. The *similarity (dissimilarity) matrix*, $P(X)$, is an $N \times N$ matrix whose (i, j) element equals the similarity $s(\boldsymbol{x}_i, \boldsymbol{x}_j)$ (dissimilarity $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$) between vectors $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. It is also referred to as the *proximity matrix* to include both cases. In general, P is a symmetric matrix.[2] Moreover, if P is a similarity matrix, its diagonal elements are equal to the maximum value of s. On the other hand, if P is a dissimilarity matrix, its diagonal elements are equal to the minimum value of d. Notice that for a single pattern matrix there exists more than one proximity matrix depending on the choice of the proximity measure $\wp(\boldsymbol{x}_i, \boldsymbol{x}_j)$. However, fixing $\wp(\boldsymbol{x}_i, \boldsymbol{x}_j)$, one can easily observe that for a given pattern matrix there exists an associated single proximity matrix. On the other hand, a proximity matrix may correspond to more than one pattern matrices (see Problem 13.1).

[1] A method that produces hierarchies, which do not, necessarily, possess the nesting property, has been proposed in [Frig 97].

[2] In [Ozaw 83] a hierarchical clustering algorithm, called RANCOR, is discussed, which is based on asymmetric proximity matrices.

Example 13.1

Let $X = \{\boldsymbol{x}_i, i = 1, \ldots, 5\}$, with $\boldsymbol{x}_1 = [1, 1]^T$, $\boldsymbol{x}_2 = [2, 1]^T$, $\boldsymbol{x}_3 = [5, 4]^T$, $\boldsymbol{x}_4 = [6, 5]^T$, and $\boldsymbol{x}_5 = [6.5, 6]^T$. The pattern matrix of X is

$$D(X) = \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 5 & 4 \\ 6 & 5 \\ 6.5 & 6 \end{bmatrix}$$

and its corresponding dissimilarity matrix, when the Euclidean distance is in use, is

$$P(X) = \begin{bmatrix} 0 & 1 & 5 & 6.4 & 7.4 \\ 1 & 0 & 4.2 & 5.7 & 6.7 \\ 5 & 4.2 & 0 & 1.4 & 2.5 \\ 6.4 & 5.7 & 1.4 & 0 & 1.1 \\ 7.4 & 6.7 & 2.5 & 1.1 & 0 \end{bmatrix}$$

When the Tanimoto measure is used, the similarity matrix of X becomes

$$P^{'}(X) = \begin{bmatrix} 1 & 0.75 & 0.26 & 0.21 & 0.18 \\ 0.75 & 1 & 0.44 & 0.35 & 0.20 \\ 0.26 & 0.44 & 1 & 0.96 & 0.90 \\ 0.21 & 0.35 & 0.96 & 1 & 0.98 \\ 0.18 & 0.20 & 0.90 & 0.98 & 1 \end{bmatrix}$$

Note that in $P(X)$ all diagonal elements are 0, since $d_2(\boldsymbol{x}, \boldsymbol{x}) = 0$, while in $P^{'}(X)$ all diagonal elements are equal to 1, since $s_T(\boldsymbol{x}, \boldsymbol{x}) = 1$.

A *threshold dendrogram*, or simply a *dendrogram*, is an effective means of representing the sequence of clusterings produced by an agglomerative algorithm. To clarify this idea, let us consider again the data set given in Example 13.1. Let us define $g(C_i, C_j)$ as $g(C_i, C_j) = d_{\min}^{ss}(C_i, C_j)$ (see Section 11.2). One may easily see that, in this case, the clustering sequence for X produced by the generalized agglomerative scheme, when the Euclidean distance between two vectors is used, is the one shown in Figure 13.1. At the first step $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ form a new cluster. At the second step $\boldsymbol{x}_4$ and $\boldsymbol{x}_5$ stick together, forming a single cluster. At the third step $\boldsymbol{x}_3$ joins the cluster $\{\boldsymbol{x}_4, \boldsymbol{x}_5\}$ and, finally, at the fourth step the clusters $\{\boldsymbol{x}_1, \boldsymbol{x}_2\}$ and $\{\boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{x}_5\}$ are merged into a single set, X. The right-hand side of Figure 13.1 shows the corresponding dendrogram. Each step of *the generalized agglomerative sheme (GAS)* corresponds to a level of the dendrogram. *Cutting the dendrogram at a specific level results in a clustering.*

A *proximity dendrogram* is a dendrogram that takes into account the level of proximity where two clusters are merged *for the first time*. When a dissimilarity (similarity) measure is in use, the proximity dendrogram is called a *dissimilarity (similarity) dendrogram*. This tool may be used as an indicator of the natural or

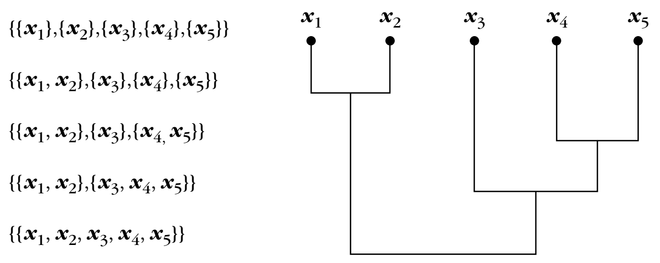

{{x_1},{x_2},{x_3},{x_4},{x_5}}
{{x_1, x_2},{x_3},{x_4},{x_5}}
{{x_1, x_2},{x_3},{x_4, x_5}}
{{x_1, x_2},{x_3, x_4, x_5}}
{{x_1, x_2, x_3, x_4, x_5}}
x_1
x_2
x_3
x_4
x_5

FIGURE 13.1

The clustering hierarchy for X of Example 13.1 and its corresponding dendrogram.

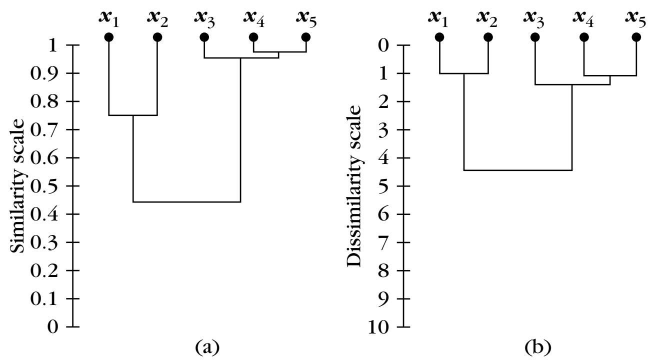


FIGURE 13.2

(a) The proximity (similarity) dendrogram for X using $P'(X)$ from Example 13.1. (b) The proximity (dissimilarity) dendrogram for X using $P(X)$ from Example 13.1.

forced formation of clusters at any level. That is, it may provide a clue about the clustering that best fits the data, as will be explained in Section 13.6. Figure 13.2 shows the similarity and dissimilarity dendrograms for X of Example 13.1 when $P'(X)$ and $P(X)$ are in use, respectively.

Before we proceed to a more detailed discussion of the hierarchical algorithms, an important note is in order. As explained earlier, this kind of algorithm determines a whole hierarchy of clusterings, rather than a single clustering. The determination of the whole dendrogram may be very useful in some applications, such as biological taxonomy (e.g., see [Prit 71]). However, in other applications we are interested only in the specific clustering that best fits the data. If one is willing to use hierarchical algorithms for applications of the latter type, he or she has to decide which clustering of the produced hierarchy is most suitable for the data. Equivalently, one must determine the appropriate level to cut the dendrogram that corresponds to the resulting hierarchy. Similar comments also hold for the divisive algorithms to be discussed later. Methods for determining the cutting level are discussed in the last section of the chapter.

In the sequel, unless otherwise stated, we consider only dissimilarity matrices. Similar arguments hold for similarity matrices.

13.2.2 Agglomerative Algorithms Based on Matrix Theory

These algorithms may be viewed as special cases of GAS. The input in these schemes is the $N \times N$ dissimilarity matrix, $P_0 = P(X)$, derived from X. At each level, t, when two clusters are merged into one, the size of the dissimilarity matrix P_t becomes $(N - t) \times (N - t)$. P_t follows from P_{t-1} by (a) deleting the two rows and columns that correspond to the merged clusters and (b) adding a new row and a new column that contain the distances between the newly formed cluster and the old (unaffected at this level) clusters. The distance between the newly formed cluster C_q (the result of merging C_i and C_j) and an old cluster, C_s, is a function of the form

$$d(C_q, C_s) = f(d(C_i, C_s), d(C_j, C_s), d(C_i, C_j)) \tag{13.2}$$

The procedure justifies the name *matrix updating algorithms*, often used in the literature. In the sequel, we give an algorithmic scheme, the *matrix updating algorithmic scheme (MUAS)*, that includes most of the algorithms of this kind. Again, t denotes the current level of the hierarchy.

Matrix Updating Algorithmic Scheme (MUAS)

```
■ Initialization:
    • $\Re_0 = \{\{\boldsymbol{x}_i\}, i = 1, \ldots, N\}$.
    • $P_0 = P(X)$.
    • $t = 0$
■ Repeat:
    • $t = t + 1$
    • Find $C_i, C_j$ such that $d(C_i, C_j) = \min_{r,s=1,\ldots,N,\ r \neq s} d(C_r, C_s)$
    • Merge $C_i$, $C_j$ into a single cluster $C_q$ and form $\Re_t = (\Re_{t-1} - \{C_i, C_j\}) \cup \{C_q\}$.
    • Define the proximity matrix $P_t$ from $P_{t-1}$ as explained in the text.
■ Until $\Re_{N-1}$ clustering is formed, that is, all vectors lie in the same cluster.
```

Notice that this scheme is in the spirit of the GAS. In [Lanc 67] it is pointed out that a number of distance functions comply with the following update equation:

$$\begin{aligned} d(C_q, C_s) = {} & a_i d(C_i, C_s) + a_j d(C_j, C_s) + b d(C_i, C_j) \\ & + c|d(C_i, C_s) - d(C_j, C_s)| \end{aligned} \tag{13.3}$$

Different values of a_i, a_j, b, and c correspond to different choices of the dissimilarity measure $d(C_i, C_j)$. Equation (13.3) is also a recursive definition of a distance between two clusters, initialized from the distance between the initial point clusters. Another formula, not involving the last term and allowing a_i, a_j, and b to be functions of C_i, C_j, and C_s, is discussed in [Bobe 93]. In the sequel we present algorithms stemming from MUAS and following from Eq. (13.3) for different values of the parameters a_i, a_j, b, c.

The simpler algorithms included in this scheme are:

- The *single link algorithm*. This is obtained from Eq. (13.3) if we set $a_i = 1/2$, $a_j = 1/2, b = 0, c = -1/2$. In this case,

$$d(C_q, C_s) = \min\{d(C_i, C_s), \quad d(C_j, C_s)\} \tag{13.4}$$

  The d_{min}^{ss} measure, defined in Section 11.2, falls under this umbrella.

- The *complete link algorithm*. This follows from Eq. (13.3) if we set $a_i = \frac{1}{2}$, $a_j = \frac{1}{2}, b = 0$ and $c = \frac{1}{2}$. Then we may write[3]

$$d(C_q, C_s) = \max\{d(C_i, C_s), \quad d(C_j, C_s)\}. \tag{13.5}$$

Note that the distance between the merged clusters C_i and C_j does not enter into the above formulas. In the case where a similarity, instead of a dissimilarity, measure is used then (a) for the single link algorithm the operator min should be replaced by max in Eq. (13.4) and (b) for the complete link algorithm the operator max should be replaced by the operator min in Eq. (13.5). To gain a further insight into the behavior of the above algorithms, let us consider the following example.

Example 13.2

Consider the data set shown in Figure 13.3a. The first seven points form an elongated cluster while the remaining four form a rather compact cluster. The numbers on top of the edges connecting the points correspond to the respective (Euclidean) distances between vectors. These distances are also taken to measure the distance between two initial point clusters. Distances that are not shown are assumed to have very large values. Figure 13.3b shows the dendrogram produced by the application of the single link algorithm to this data set. As one can easily observe, the algorithm first recovers the elongated cluster, and the second cluster is recovered at a higher dissimilarity level.

Figure 13.3c shows the dendrogram produced by the complete link algorithm. It is easily noticed that this algorithm proceeds by recovering first compact clusters.

[3] Equations (13.4) and (13.5) suggest that merging clusters is a min/max problem for the complete link and a min/min problem for the single link.

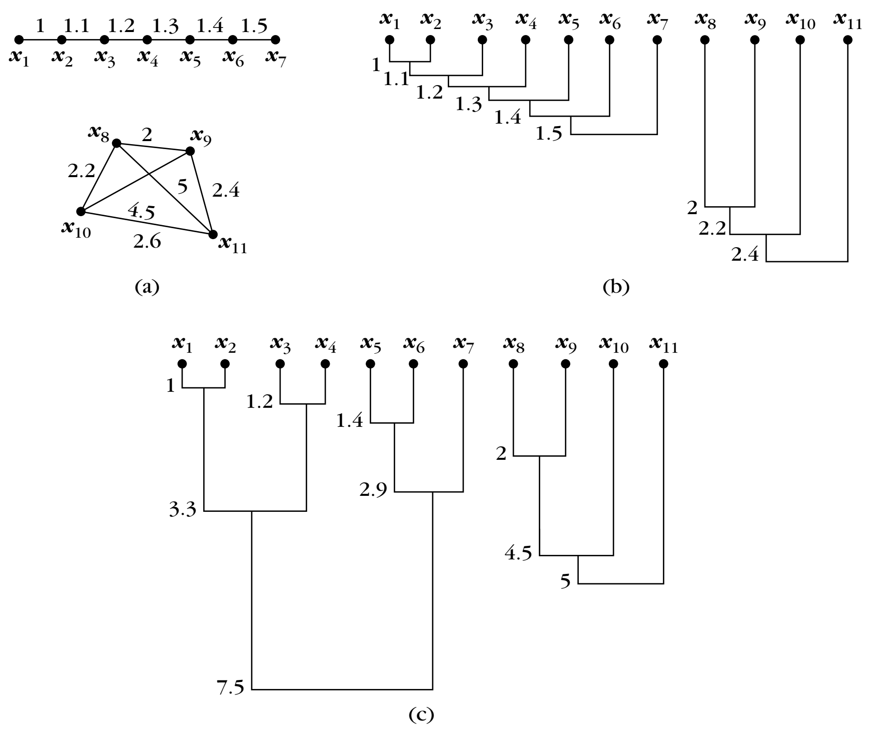

1 1.1 1.2 1.3 1.4 1.5
x_1 x_2 x_3 x_4 x_5 x_6 x_7
x_8 2 x_9
2.2 5 2.4 4.5
x_{10} 2.6 x_{11}
(a)
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10} x_{11}
1 1.1 1.2 1.3 1.4 1.5
2 2.2 2.4
(b)
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10} x_{11}
1 1.2 1.4 2 2.9 3.3 4.5 5 7.5
(c)

FIGURE 13.3

(a) The data set X. (b) The dissimilarity dendrogram produced by the single link algorithm. (c) The dissimilarity dendrogram produced by the complete link algorithm (the level of the final clustering is not shown).

Remark

- The preceding algorithms are the two extremes of the family described by Eq. (13.3). Indeed, the clusters produced by the single link algorithm are formed at low dissimilarities in the dissimilarity dendrogram. On the other hand, the clusters produced by the complete link algorithm are formed at high dissimilarities in the dissimilarity dendrogram. This happens because in the single link (complete link) algorithm the minimum (maximum) of the distances $d(C_i, C_s)$ and $d(C_j, C_s)$ is used as the distance between $d(C_q, C_s)$. This implies that the single link algorithm has a tendency to favor elongated clusters. This characteristic is also known as the *chaining effect*. On the other hand, the complete link algorithm proceeds by recovering small compact clusters, and it should be preferred if there is evidence that compact clusters underlie X.

The rest of the algorithms, to be discussed next, are compromises between these two extremes.[4]

- The *weighted pair group method average (WPGMA)* algorithm is obtained from Eq. (13.3) if we set $a_i = a_j = \frac{1}{2}, b = 0$, and $c = 0$, that is,

$$d(C_q, C_s) = \frac{1}{2}(d(C_i, C_s) + d(C_j, C_s)) \tag{13.6}$$

  Thus, in this case the distance between the newly formed cluster C_q and an old one C_s is defined as the average of distances between C_i, C_s and C_j, C_s.

- The *unweighted pair group method average (UPGMA)* algorithm is defined if we choose $a_i = \frac{n_i}{n_i+n_j}, a_j = \frac{n_j}{n_i+n_j}, b = 0, c = 0$, where n_i and n_j are the cardinalities of C_i and C_j, respectively. In this case the distance between C_q and C_s is defined as

$$d(C_q, C_s) = \frac{n_i}{n_i + n_j} d(C_i, C_s) + \frac{n_j}{n_i + n_j} d(C_j, C_s) \tag{13.7}$$

- The *unweighted pair group method centroid (UPGMC)* algorithm results on setting $a_i = \frac{n_i}{n_i+n_j}, a_j = \frac{n_j}{n_i+n_j}, b = -\frac{n_i n_j}{(n_i+n_j)^2}, c = 0$, that is,

$$d_{qs} = \frac{n_i}{n_i + n_j} d_{is} + \frac{n_j}{n_i + n_j} d_{js} - \frac{n_i n_j}{(n_i + n_j)^2} d_{ij} \tag{13.8}$$

  where d_{qs} has been used in place of $d(C_q, C_s)$ for notational simplicity. This algorithm has an interesting interpretation. Let the representatives of the clusters be chosen as the respective means (centroids), that is,

$$\boldsymbol{m}_q = \frac{1}{n_q} \sum_{\boldsymbol{x} \in C_q} \boldsymbol{x} \tag{13.9}$$

  and the dissimilarity to be the squared Euclidean distance between cluster representatives. Then it turns out that this recursive definition of d_{qs} is nothing but the square Euclidean distance between the respective representatives (see Problem 13.2), that is,

$$d_{qs} = \|\boldsymbol{m}_q - \boldsymbol{m}_s\|^2 \tag{13.10}$$

- The *weighted pair group method centroid (WPGMC)* algorithm is obtained if we choose $a_i = a_j = \frac{1}{2}, b = -\frac{1}{4}$, and $c = 0$. That is,

$$d_{qs} = \frac{1}{2} d_{is} + \frac{1}{2} d_{js} - \frac{1}{4} d_{ij} \tag{13.11}$$

[4] The terminology used here follows that given in [Jain 88].

Note that Eq. (13.11) results from (13.8) if the merging clusters have the same number of vectors. Of course, this is not true in general, and the algorithm basically computes the distance between weighted versions of the respective centroids. A notable feature of the WPGMC algorithm is that there are cases where $d_{qs} \leq \min(d_{is}, d_{js})$ (Problem 13.3).

- The *Ward or minimum variance algorithm*. Here, the distance between two clusters C_i and C_j, d'_{ij}, is defined as a weighted version of the squared Euclidean distance of their mean vectors, that is,

$$d'_{ij} = \frac{n_i n_j}{n_i + n_j} d_{ij} \tag{13.12}$$

where $d_{ij} = \|\boldsymbol{m}_i - \boldsymbol{m}_j\|^2$. Thus, in step 2.2 of MUAS we seek the pair of clusters C_i, C_j so that the quantity d'_{ij} is minimum. Furthermore, it can be shown (Problem 13.4) that this distance belongs to the family of Eq. (13.3), and we can write

$$d'_{qs} = \frac{n_i + n_s}{n_i + n_j + n_s} d'_{is} + \frac{n_j + n_s}{n_i + n_j + n_s} d'_{js} - \frac{n_s}{n_i + n_j + n_s} d'_{ij} \tag{13.13}$$

The preceding distance can also be viewed from a different perspective. Let us define

$$e_r^2 = \sum_{\boldsymbol{x} \in C_r} \|\boldsymbol{x} - \boldsymbol{m}_r\|^2$$

as the variance of the rth cluster around its mean and

$$E_t = \sum_{r=1}^{N-t} e_r^2 \tag{13.14}$$

as the total variance of the clusters at the tth level (where $N - t$ clusters are present). We will now show that *Ward's algorithm forms $\Re_{t+1}$ by merging the two clusters that lead to the smallest possible increase of the total variance.* Suppose that clusters C_i and C_j are chosen to be merged into one, say C_q. Let E_{t+1}^{ij} be the total variance after the clusters C_i and C_j are merged in C_q at the $t + 1$ level. Then, since all other clusters remain unaffected, the difference $\Delta E_{t+1}^{ij} = E_{t+1}^{ij} - E_t$ is equal to

$$\Delta E_{t+1}^{ij} = e_q^2 - e_i^2 - e_j^2 \tag{13.15}$$

Taking into account that

$$\sum_{\boldsymbol{x} \in C_r} \|\boldsymbol{x} - \boldsymbol{m}_r\|^2 = \sum_{\boldsymbol{x} \in C_r} \|\boldsymbol{x}\|^2 - n_r \|\boldsymbol{m}_r\|^2 \tag{13.16}$$

Eq. (13.15) is written as

$$\Delta E_{t+1}^{ij} = n_i \|\boldsymbol{m}_i\|^2 + n_j \|\boldsymbol{m}_j\|^2 - n_q \|\boldsymbol{m}_q\|^2 \tag{13.17}$$

Using the fact that

$$n_i \boldsymbol{m}_i + n_j \boldsymbol{m}_j = n_q \boldsymbol{m}_q \tag{13.18}$$

Eq. (13.17) becomes

$$\Delta E_{t+1}^{ij} = \frac{n_i n_j}{n_i + n_j} \|\boldsymbol{m}_i - \boldsymbol{m}_j\|^2 = d_{ij}^{'} \tag{13.19}$$

which is the distance minimized by Ward's algorithm. This justifies the name minimum variance.

Example 13.3

Consider the following dissimilarity matrix:

$$P_0 = \begin{bmatrix} 0 & 1 & 2 & 26 & 37 \\ 1 & 0 & 3 & 25 & 36 \\ 2 & 3 & 0 & 16 & 25 \\ 26 & 25 & 16 & 0 & 1.5 \\ 37 & 36 & 25 & 1.5 & 0 \end{bmatrix}$$

where the corresponding squared Euclidean distance is adopted. As one can easily observe, the first three vectors, $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, and $\boldsymbol{x}_3$, are very close to each other and far away from the others. Likewise, $\boldsymbol{x}_4$ and $\boldsymbol{x}_5$ lie very close to each other and far away from the first three vectors. For this problem all seven algorithms discussed before result in the same dendrogram. The only difference is that each clustering is formed at a different dissimilarity level.

Let us first consider the single link algorithm. Since P_0 is symmetric, we consider only the upper diagonal elements. The smallest of these elements equals 1 and occurs at position $(1, 2)$ of P_0. Thus, $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ come into the same cluster and $\Re_1 = \{\{\boldsymbol{x}_1, \boldsymbol{x}_2\}, \{\boldsymbol{x}_3\}, \{\boldsymbol{x}_4\}, \{\boldsymbol{x}_5\}\}$ is produced. In the sequel, the dissimilarities among the newly formed cluster and the remaining ones have to be computed. This can be achieved via Eq. (13.4). The resulting proximity matrix, P_1, is

$$P_1 = \begin{bmatrix} 0 & 2 & 25 & 36 \\ 2 & 0 & 16 & 25 \\ 25 & 16 & 0 & 1.5 \\ 36 & 25 & 1.5 & 0 \end{bmatrix}$$

Its first row and column correspond to the cluster $\{\boldsymbol{x}_1, \boldsymbol{x}_2\}$. The smallest of the upper diagonal elements of P_1 equals 1.5. This means that at the next stage, the clusters $\{\boldsymbol{x}_4\}$ and $\{\boldsymbol{x}_5\}$ will stick together into a single cluster, producing $\Re_2 = \{\{\boldsymbol{x}_1, \boldsymbol{x}_2\}, \{\boldsymbol{x}_3\}, \{\boldsymbol{x}_4, \boldsymbol{x}_5\}\}$. Employing Eq. (13.4), we obtain

$$P_2 = \begin{bmatrix} 0 & 2 & 25 \\ 2 & 0 & 16 \\ 25 & 16 & 0 \end{bmatrix}$$

where the first row (column) corresponds to $\{\boldsymbol{x}_1, \boldsymbol{x}_2\}$, and the second and third rows (columns) correspond to $\{\boldsymbol{x}_3\}$ and $\{\boldsymbol{x}_4, \boldsymbol{x}_5\}$, respectively. Proceeding as before, at the next stage $\{\boldsymbol{x}_1, \boldsymbol{x}_2\}$

and $\{\boldsymbol{x}_3\}$ will get together in a single cluster and $\Re_3 = \{\{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3\}, \{\boldsymbol{x}_4, \boldsymbol{x}_5\}\}$ is produced. The new proximity matrix, P_3, becomes

$$P_3 = \begin{bmatrix} 0 & 16 \\ 16 & 0 \end{bmatrix}$$

where the first and the second row (column) correspond to $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3\}$ and $\{\boldsymbol{x}_4, \boldsymbol{x}_5\}$ clusters, respectively. Finally, $\Re_4 = \{\{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{x}_5\}\}$ will be formed at dissimilarity level equal to 16.

Working in a similar fashion, we can apply the remaining six algorithms to P_0. Note that in the case of Ward's algorithm, the initial dissimilarity matrix should be $\frac{1}{2}P_0$, due to the definition in Eq. (13.12). However, care must be taken when we apply UPGMA, UPGMC, and Ward's method. In these cases, when a merging takes place the parameters a_i, a_j, b, and c must be properly adjusted. The proximity levels at which each clustering is formed for each algorithm are shown in Table 13.1.

The considered task is a nice problem with two well-defined compact clusters lying away from each other. The preceding example demonstrates that in such "easy" cases all algorithms work satisfactorily (as happens with most of the clustering algorithms proposed in the literature). The particular characteristics of each algorithm are revealed when more demanding situations are faced. Thus, in Example 13.2, we saw the different behaviors of the single link and complete link algorithms. Characteristics of other algorithms, such as the WPGMC and the UPGMC, are discussed next.

13.2.3 Monotonicity and Crossover

Let us consider the following dissimilarity matrix:

$$P = \begin{bmatrix} 0 & 1.8 & 2.4 & 2.3 \\ 1.8 & 0 & 2.5 & 2.7 \\ 2.4 & 2.5 & 0 & 1.2 \\ 2.3 & 2.7 & 1.2 & 0 \end{bmatrix}$$

Table 13.1 The Results Obtained with the Seven Algorithms Discussed when they are Applied to the Proximity Matrix of Example 13.3

	SL	CL	WPGMA	UPGMA	WPGMC	UPGMC	Ward's Algorithm
$\Re_0$	0	0	0	0	0	0	0
$\Re_1$	1	1	1	1	1	1	0.5
$\Re_2$	1.5	1.5	1.5	1.5	1.5	1.5	0.75
$\Re_3$	2	3	2.5	2.5	2.25	2.25	1.5
$\Re_4$	16	37	25.75	27.5	24.69	26.46	31.75

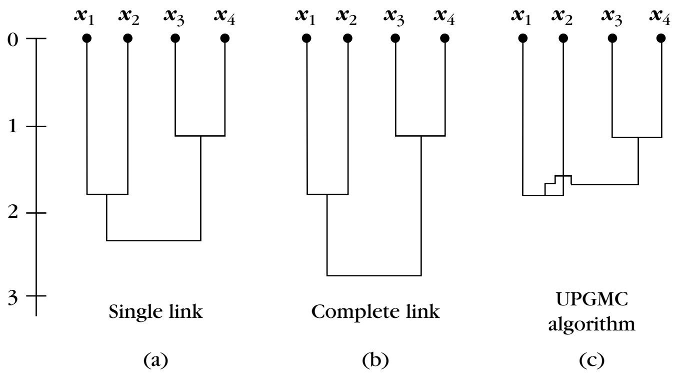


FIGURE 13.4

Dissimilarity dendrograms derived from (a) single link, (b) complete link, and (c) UPGMC and WPGMC when they apply to P. The third dendrogram exhibits the crossover phenomenon.

Application of the single and complete link algorithms to P gives rise to the dissimilarity dendrograms depicted in Figures 13.4a and 13.4b, respectively. Application of the UPGMC and WPGMC algorithms to P results in the same dissimilarity dendrogram, which is shown in Figure 13.4c. In this dendrogram something interesting occurs. The cluster $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4\}$ is formed at a lower dissimilarity level than cluster $\{\boldsymbol{x}_1, \boldsymbol{x}_2\}$. This phenomenon is called *crossover*. More specifically, crossover occurs when a cluster is formed at a lower dissimilarity level than any of its components. The opposite of the crossover is *monotonicity*. Satisfaction of the latter condition implies that each cluster is formed at a higher dissimilarity level than any one of its components. More formally, the monotonicity condition may be stated as follows:

"If clusters C_i and C_j are selected to be merged in cluster C_q, at the tth level of the hierarchy, then the following condition must hold:

$$d(C_q, C_k) \geq d(C_i, C_j)$$

for all $C_k, k \neq i, j, q$."

Monotonicity is a property that is exclusively related to the clustering algorithm and not to the (initial) proximity matrix.

Recall Eq. (13.3) defined in terms of the parameters a_i, a_j, b, and c. In the sequel, a proposition is stated and proved that allows us to decide whether an algorithm satisfies the monotonicity condition.

Proposition 1. *If a_i and a_j are nonnegative, $a_i + a_j + b \geq 1$, and either (a) $c \geq 0$ or (b) $\max\{-a_i, -a_j\} \leq c \leq 0$, then the corresponding clustering method satisfies the monotonicity condition.*

Proof. (a) From the hypothesis we have that

$$b \geq 1 - a_i - a_j$$

Substituting this result in Eq. (13.3) and after some rearrangements, we obtain

$$\begin{aligned} d(C_q, C_s) \geq\ & d(C_i, C_j) + a_i(d(C_i, C_s) - d(C_i, C_j)) \\ & + a_j(d(C_j, C_s) - d(C_i, C_j)) + c|d(C_i, C_s) - d(C_j, C_s)| \end{aligned}$$

Since, from step 2.2 of the MUAS in Section 13.2.2,

$$d(C_i, C_j) = \min_{r,u} d(C_r, C_u)$$

the second and third terms of the last inequality are nonnegative. Moreover, the fourth term of the same inequality is also nonnegative. Therefore, we obtain

$$d(C_q, C_s) \geq d(C_i, C_j)$$

Thus, the monotonicity condition is satisfied.

(b) Let $d(C_i, C_s) \geq d(C_j, C_s)$ (the case where $d(C_i, C_s) < d(C_j, C_s)$ may be treated similarly). As in the previous case,

$$b \geq 1 - a_i - a_j$$

Taking into account this inequality, Eq. (13.3) gives

$$\begin{aligned} d(C_q, C_s) \geq\ & d(C_i, C_j) + a_i(d(C_i, C_s) - d(C_i, C_j)) \\ & + a_j(d(C_j, C_s) - d(C_i, C_j)) + c(d(C_i, C_s) - d(C_j, C_s)) \end{aligned}$$

By adding and subtracting on the right-hand side of this inequality, the term $cd(C_i, C_j)$ and after some manipulations, we obtain

$$\begin{aligned} d(C_q, C_s) \geq\ & (a_j - c)(d(C_j, C_s) - d(C_i, C_j)) + d(C_i, C_j) \\ & + (a_i + c)(d(C_i, C_s) - d(C_i, C_j)) \end{aligned}$$

Since, from the hypothesis, $a_j - c \geq 0$ and

$$d(C_i, C_j) = \min_{r,u} d(C_r, C_u)$$

from step 2.2 of the MUAS we obtain that

$$d(C_q, C_s) \geq d(C_i, C_j) \qquad \square$$

Note that the conditions of Proposition 1 are sufficient but not necessary. This means that algorithms that do not satisfy the premises of this proposition may still satisfy the monotonicity condition. It is easy to note that the single link, the complete link, the UPGMA, the WPGMA, and Ward's algorithm satisfy the premises of

Proposition 1. Thus, all these algorithms satisfy the monotonicity condition. The other two algorithms, the UPGMC and the WPGMC, do not satisfy the monotonicity condition. Moreover, we can construct examples that demonstrate that these two algorithms violate the monotonicity property, as follows from Figure 13.4c. However, it is this does not mean that they always lead to dendrograms with crossovers.

Finally, we note that there have been several criticisms concerning the usefulness of algorithms that do not satisfy the monotonicity condition (e.g., [Will 77, Snea 73]). However, these algorithms may give satisfactory results in the frame of certain applications. Moreover, there is no theoretical guideline suggesting that the algorithms satisfying the monotonicity condition always lead to acceptable results. After all, this ought to be the ultimate criterion for the usefulness of an algorithm (unfortunately, such a criterion does not exist in general).

13.2.4 Implementational Issues

As stated earlier, the computational time of GAS is $O(N^3)$. However, many efficient implementations of these schemes have been proposed in the literature, which reduce the computational time by an order of N. For example, in [Kuri 91] an implementation is discussed, for which the required computational time is reduced to $O(N^2 \log N)$. Also, in [Murt 83, Murt 84, Murt 85], implementations for widely used agglomerative algorithms are discussed that require $O(N^2)$ computational time and either $O(N^2)$ or $O(N)$ storage. Finally, parallel implementations on single instruction multiple data (SIMD) machines are discussed in [Will 89] and [Li 90].

13.2.5 Agglomerative Algorithms Based on Graph Theory

Before we describe the algorithms of this family, let us first recall some basic definitions from graph theory.

Basic Definitions from Graph Theory

A *graph* G is defined as an ordered pair $G = (V, E)$, where $V = \{v_i,\ i = 1, \ldots, N\}$ is a set of *vertices* and E is a set of *edges* connecting some pairs of vertices. An edge, connecting the vertices v_i and v_j, will be denoted either by e_{ij} or by (v_i, v_j). When the ordering of v_i and v_j is of no importance, then we deal with *undirected graphs*. Otherwise, we deal with *directed graphs*. In addition, if no cost is associated with the edges of the graph, we deal with *unweighted graphs*. Otherwise, we deal with *weighted graphs*. In the sequel, we consider graphs where a pair of vertices may be connected by at most one edge. In the framework of clustering, we deal with undirected graphs where each vertex corresponds to a feature vector (or, equivalently, to the pattern represented by the feature vector).

A *path* in G, between vertices v_{i_1} and v_{i_n}, is a sequence of vertices and edges of the form $v_{i_1} e_{i_1 i_2} v_{i_2} \ldots v_{i_{n-1}} e_{i_{n-1} i_n} v_{i_n}$ (Figure 13.5a). Of course, there is no guaranteed, that there will be always exists a path from v_{i_1} to v_{i_n}. If in this path, v_{i_1}

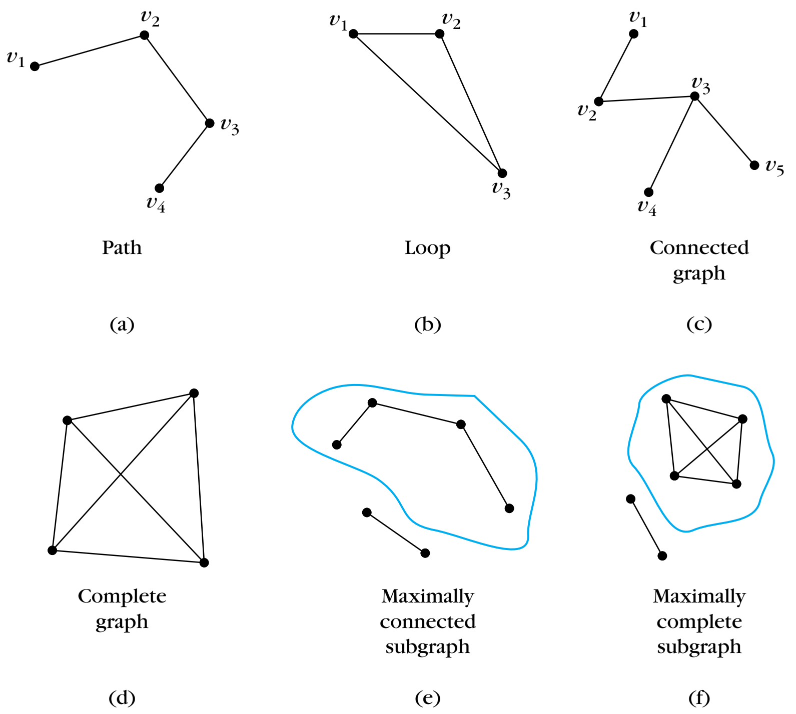


FIGURE 13.5

(a) A path connecting the vertices v_1 and v_4. (b) A loop. (c) A connected graph. (d) A complete graph. (e) Maximally connected subgraph. (f) Maximally complete subgraph.

coincides with v_{i_n}, the path is called a *loop* or *circle* (Figure 13.5b). In the special case in which an edge connects a vertex to itself, we have a *self-loop*.

A *subgraph* $G' = (V', E')$ of G is a graph with $V' \subseteq V$ and $E' \subseteq E_1$, where E_1 is a subset of E, whose edges connect vertices that lie in V'. Clearly, G is a subgraph of itself.

A *subgraph* $G' = (V', E')$ is *connected* if there exists at least one path connecting any pair of vertices in V' (Figure 13.5c). For example, in Figure 13.5c the subgraph with vertices v_1, v_2, v_4, and v_5 is not connected. The subgraph G' is *complete* if every vertex $v_i \in V'$ is connected with every vertex in $V' - \{v_i\}$ (Figure 13.5d).

A *maximally connected subgraph of* G is a connected subgraph G' of G that contains as many vertices of G as possible (Figure 13.5e). A *maximally complete subgraph* is a complete subgraph G' of G that contains as many vertices of G as possible (Figure 13.5f).

A concept that is closely related to the algorithms based on graph theory is that of the *threshold graph*. A threshold graph is an undirected, unweighted graph with

N nodes, each corresponding to a vector of the data set X. In this graph there are no self-loops or multiple edges between any two nodes. Let a be a dissimilarity level. A threshold graph $G(a)$ with N nodes contains an edge between two nodes i and j *if the dissimilarity between the corresponding vectors* $\boldsymbol{x}_i$ *and* $\boldsymbol{x}_j$ *is less than or equal to* $a, i,j = 1, \ldots, N$. Alternatively, we may write

$$(v_i, v_j) \in G(a), \quad \text{if } d(\boldsymbol{x}_i, \boldsymbol{x}_j) \leq a, \quad i,j = 1, \ldots, N \tag{13.20}$$

If similarity measures are used, this definition is altered to

$$(v_i, v_j) \in G(a), \quad \text{if } s(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq a, \quad i,j = 1, \ldots, N$$

A *proximity graph* $G_p(a)$ is a threshold graph $G(a)$, all of whose edges (v_i, v_j) are weighted with the proximity measure between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. If a dissimilarity (similarity) measure is used as proximity between two vectors, then the proximity graph is called a *dissimilarity (similarity) graph*. Figure 13.6 shows the threshold and proximity graphs $G(3)$ and $G_p(3)$, $G(5)$ and $G_p(5)$ obtained from the dissimilarity matrix $P(X)$ given in Example 13.1.

The Algorithms

In this section, we discuss agglomerative algorithms based on graph theory concepts. More specifically, we consider graphs, G, of N nodes with each node corresponding to a vector of X. Clusters are formed by connecting nodes together, and this leads to connected subgraphs. Usually, an additional graph property, $h(k)$, must be satisfied by the subgraphs in order to define valid clusters. In this context, the function g involved in GAS is replaced by $g_{h(k)}$, where $h(k)$ is the graph property. Some typical properties that can be adopted are [Jain 88, Ling 72].

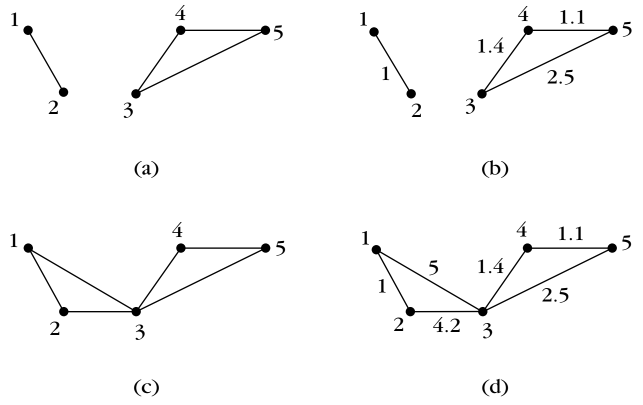


FIGURE 13.6

(a) The threshold graph $G(3)$, (b) the proximity (dissimilarity) graph $G_p(3)$, (c) the threshold graph $G(5)$, (d) the proximity (dissimilarity) graph $G_p(5)$, obtained from the dissimilarity matrix $P(X)$ of Example 13.1.

- *Node connectivity.* The node connectivity of a connected subgraph is the largest integer k such that all pairs of nodes are joined by at least k paths having no nodes in common.
- *Edge connectivity.* The edge connectivity of a connected subgraph is the largest integer k such that all pairs of nodes are joined by at least k paths having no edges in common.
- *Node degree.* The degree of a connected subgraph is the largest integer k such that each node has at least k incident edges.

The general agglomerative scheme in the context of graph theory is known as the *graph theory-based algorithmic scheme (GTAS)*. The algorithm follows exactly the same iteration steps as the generalized agglomerative scheme (GAS), with the exception of step 2.2. This is expressed as

$$g_{h(k)}(C_i, C_j) = \begin{cases} \min_{r,s} g_{h(k)}(C_r, C_s), & \text{for dissimilarity functions} \\ \max_{r,s} g_{h(k)}(C_r, C_s), & \text{for similarity functions} \end{cases} \tag{13.21}$$

The proximity function $g_{h(k)}(C_r, C_s)$ between two clusters is defined in terms of (a) a proximity measure between vectors (that is nodes in the graph) *and* (b) certain constraints imposed by the property $h(k)$ on the subgraphs that are formed. In a more formal way, $g_{h(k)}$ is defined as

$$\begin{aligned} &g_{h(k)}(C_r, C_s) \\ &\quad = \min_{\boldsymbol{x}_u \in C_r, \boldsymbol{x}_v \in C_s} \{d(\boldsymbol{x}_u, \boldsymbol{x}_v) \equiv a : \text{ the } G(a) \text{ subgraph} \\ &\qquad\qquad \text{defined by } C_r \cup C_s \text{ is (a) connected and either} \\ &\qquad\qquad \text{(b1) has the property } h(k) \text{ or (b2) is complete}\}^{5} \end{aligned} \tag{13.22}$$

In words, clusters (that is connected subgraphs) are merged (a) based on the proximity measure between their nodes and (b) provided that their merging leads to a connected subgraph that either has property $h(k)$ or is complete. Let us now consider a few examples.

Single Link Algorithm

Here connectedness is the only prerequisite. That is, no property $h(k)$ is imposed and no completeness is required. Thus (b1) and (b2) in (13.22) are ignored and (13.22) is simplified to

$$\begin{aligned} g_{h(k)}(C_r, C_s) = \min_{\boldsymbol{x}_u \in C_r, \boldsymbol{x}_v \in C_s} &\{d(\boldsymbol{x}_u, \boldsymbol{x}_v) \equiv a\text{: the } G(a) \\ &\text{subgraph defined by } C_r \cup C_s \text{ is connected}\} \end{aligned} \tag{13.23}$$

Let us demonstrate the algorithm via an example.

[5] This means that *all* nodes of $C_r \cup C_s$ participate in the required properties.

Example 13.4

Consider the following dissimilarity matrix:

$$P = \begin{bmatrix} 0 & 1.2 & 3 & 3.7 & 4.2 \\ 1.2 & 0 & 2.5 & 3.2 & 3.9 \\ 3 & 2.5 & 0 & 1.8 & 2.0 \\ 3.7 & 3.2 & 1.8 & 0 & 1.5 \\ 4.2 & 3.9 & 2.0 & 1.5 & 0 \end{bmatrix}$$

The first clustering, $\Re_0$, is the one where each vector of X forms a single cluster (see Figure 13.7). In order to determine the next clustering, $\Re_1$, via the single link algorithm, we need to compute $g_{h(k)}(C_r, C_s)$ for all pairs of the existing clusters. For $\{\boldsymbol{x}_1\}$ and $\{\boldsymbol{x}_2\}$, the value of $g_{h(k)}$ is equal to 1.2, since $\{\boldsymbol{x}_1\} \cup \{\boldsymbol{x}_2\}$ become connected for a first time in $G(1.2)$. Likewise, $g_{h(k)}(\{\boldsymbol{x}_1\}, \{\boldsymbol{x}_3\}) = 3$. The rest of the $g_{h(k)}$ values are computed in a similar fashion. Then, using Eq. (13.21), we find that $g_{h(k)}(\{\boldsymbol{x}_1\}, \{\boldsymbol{x}_2\}) = 1.2$ is the minimum value of $g_{h(k)}$ and thus $\{\boldsymbol{x}_1\}$ and $\{\boldsymbol{x}_2\}$ are merged in order to produce

$$\Re_1 = \{\{\boldsymbol{x}_1, \boldsymbol{x}_2\}, \{\boldsymbol{x}_3\}, \{\boldsymbol{x}_4\}, \{\boldsymbol{x}_5\}\}$$

Following the same procedure, we find that the minimum $g_{h(k)}$ among all pairs of clusters is $g_{h(k)}(\{\boldsymbol{x}_4\}, \{\boldsymbol{x}_5\}) = 1.5$. Thus, $\Re_2$ is given by

$$\Re_2 = \{\{\boldsymbol{x}_1, \boldsymbol{x}_2\}, \{\boldsymbol{x}_3\}, \{\boldsymbol{x}_4, \boldsymbol{x}_5\}\}$$

For the formation of $\Re_3$ we first consider the clusters $\{\boldsymbol{x}_3\}$ and $\{\boldsymbol{x}_4, \boldsymbol{x}_5\}$. In this case $g_{h(k)}(\{\boldsymbol{x}_3\}, \{\boldsymbol{x}_4, \boldsymbol{x}_5\}) = 1.8$, since $\{\boldsymbol{x}_3\} \cup \{\boldsymbol{x}_4, \boldsymbol{x}_5\}$ becomes connected at $G(1.8)$ for a first time. Similarly, we find that $g_{h(k)}(\{\boldsymbol{x}_1, \boldsymbol{x}_2\}, \{\boldsymbol{x}_3\}) = 2.5$, and $g_{h(k)}(\{\boldsymbol{x}_1, \boldsymbol{x}_2\}, \{\boldsymbol{x}_4, \boldsymbol{x}_5\}) = 3.2$. Thus,

$$\Re_3 = \{\{\boldsymbol{x}_1, \boldsymbol{x}_2\}, \{\boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{x}_5\}\}$$

Finally, we find that $g_{h(k)}(\{\boldsymbol{x}_1, \boldsymbol{x}_2\}, \{\boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{x}_5\}) = 2.5$ and $\Re_4$ is formed at this level. Observe that at $G(2.0)$ no clustering is formed.

Remark

- In the single link algorithm no property $h(k)$ is required, and Eq. (13.23) is basically the same as

$$g_{h(k)}(C_r, C_s) = \min_{x \in C_r, y \in C_s} d(\boldsymbol{x}, \boldsymbol{y}) \tag{13.24}$$

  Hence the algorithm is equivalent to its single link counterpart based on matrix theory and both produce the same results (see Problem 13.7).

Complete Link Algorithm

The only prerequisite here is that of completeness; that is, the graph property $h(k)$ is omitted. Since, connectedness is a weaker condition than completeness, subgraphs form valid clusters only if they are complete. Let us demonstrate the algorithms through the case of Example 13.4.

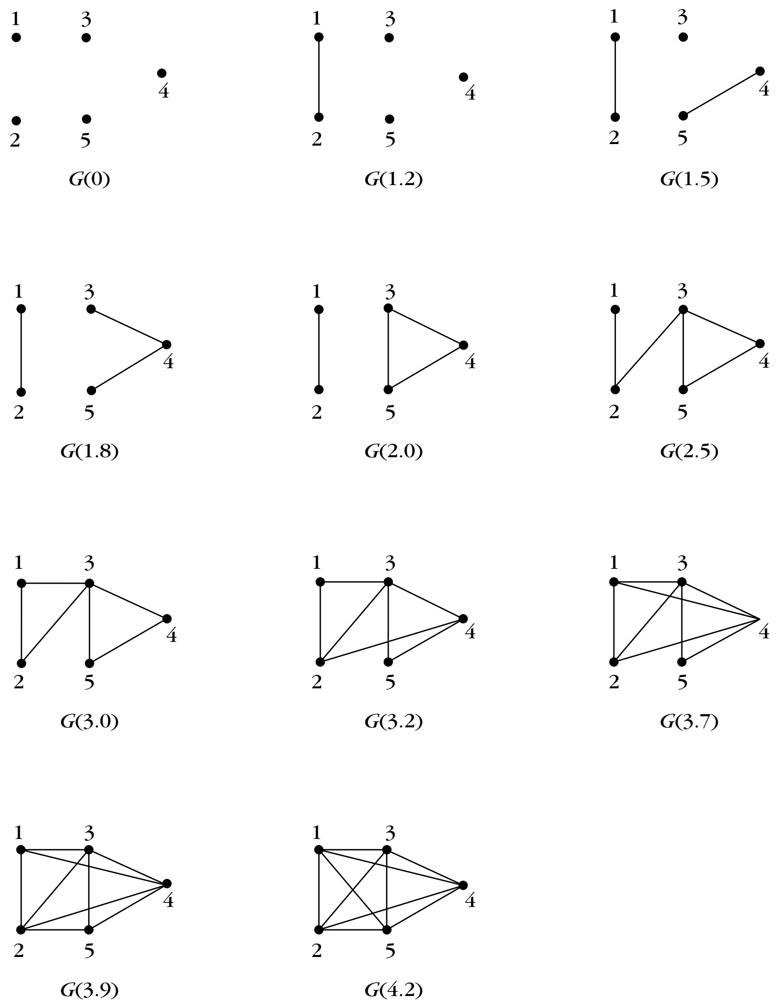


FIGURE 13.7

Threshold graphs derived by the dissimilarity matrix $\boldsymbol{P}$ given in Example 13.4.

Clusterings $\Re_0, \Re_1$, and $\Re_2$ are the same as those produced by the single link algorithm and are formed by $G(0), G(1.2)$, and $G(1.5)$, respectively. Let us derive the $\Re_3$ clustering. It is $g_{h(k)}(\{\boldsymbol{x}_3\}, \{\boldsymbol{x}_4, \boldsymbol{x}_5\}) = 2$, because at $G(2.0)$, $\{\boldsymbol{x}_3\} \cup \{\boldsymbol{x}_4, \boldsymbol{x}_5\}$ becomes complete for the first time. Similarly, $g_{h(k)}(\{\boldsymbol{x}_1, \boldsymbol{x}_2\}, \{\boldsymbol{x}_3\}) = 3$ and $g_{h(k)}(\{\boldsymbol{x}_1, \boldsymbol{x}_2\}, \{\boldsymbol{x}_4, \boldsymbol{x}_5\}) = 4.2$. Thus, the resulting $\Re_3$ clustering is the same as the one obtained by the single link algorithm. The only difference is that it is formed at graph $G(2.0)$ instead of $G(1.8)$, which was the case with the single link algorithm. Finally, the last clustering, $\Re_4$ is defined at $G(4.2)$.

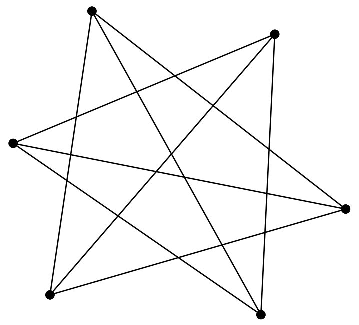


FIGURE 13.8

A graph with node connectivity, edge connectivity, and node degree equal to 3.

Remark

- A little thought suffices to see that Eq. (13.22) for the complete link algorithm is equivalent to

$$g_{h(k)}(C_r, C_s) = \max_{\boldsymbol{x} \in C_r, \boldsymbol{y} \in C_s} d(\boldsymbol{x}, \boldsymbol{y}) \tag{13.25}$$

  and, thus, this algorithm is equivalent to its matrix-based counterpart (see Problem 13.8)

The single and the complete link algorithms may be seen as the extreme cases of the GTAS scheme. This is because the criteria adopted for the formation of a new cluster are the weakest possible for the single link and the strongest possible for the complete link algorithm. A variety of algorithms between these two extremes may be obtained if we make different choices for $g_{h(k)}$. From Eq. (13.22) it is clear that this may be achieved by changing the property $h(k)$, where k is a parameter whose meaning depends on the adopted property $h(k)$. For example, in Figure 13.8, the value of k for the properties of node connectivity, edge connectivity, and node degree is 3.

Example 13.5

In this example we demonstrate the operation of the property $h(k)$. Let us consider the following dissimilarity matrix:

$$P(X) = \begin{bmatrix} 0 & 1 & 9 & 18 & 19 & 20 & 21 \\ 1 & 0 & 8 & 13 & 14 & 15 & 16 \\ 9 & 8 & 0 & 17 & 10 & 11 & 12 \\ 18 & 13 & 17 & 0 & 5 & 6 & 7 \\ 19 & 14 & 10 & 5 & 0 & 3 & 4 \\ 20 & 15 & 11 & 6 & 3 & 0 & 2 \\ 21 & 16 & 12 & 7 & 4 & 2 & 0 \end{bmatrix}$$

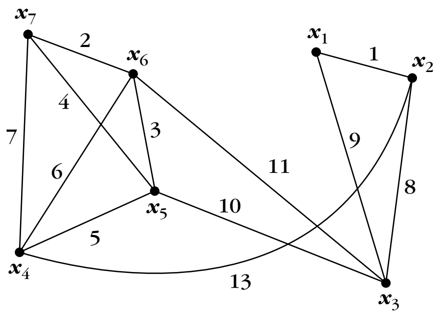


FIGURE 13.9

The proximity graph $G(13)$ derived by the dissimilarity matrix P given in Example 13.5.

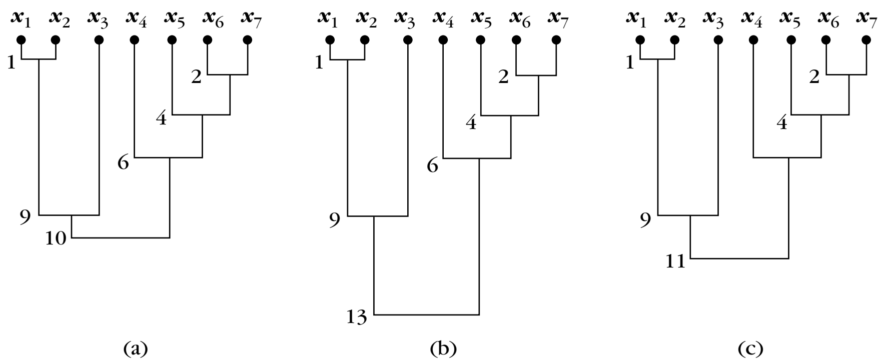


FIGURE 13.10

Dissimilarity dendrograms related to Example 13.5. (a) Dissimilarity dendrogram produced when $h(k)$ is the node degree property, with $k = 2$. (b) Dissimilarity dendrogram produced when $h(k)$ is the node connectivity property, with $k = 2$. (c) Dissimilarity dendrogram produced when $h(k)$ is the edge connectivity property, with $k = 2$.

Figure 13.9 shows the $G(13)$ proximity graph produced by this dissimilarity matrix. Let $h(k)$ be the node degree property with $k = 2$; that is, it is required that each node has at least two incident edges. Then the obtained threshold dendrogram is shown in Figure 13.10a. At dissimilarity level 1, x_1 and x_2 form a single cluster. This happens because $\{x_1\} \cup \{x_2\}$ is complete at $G(1)$, despite the fact that property $h(2)$ is not satisfied (remember the disjunction between conditions (b1) and (b2) in Eq. (13.22)). Similarly, $\{x_6\} \cup \{x_7\}$ forms a cluster at dissimilarity level 2. The next clustering is formed at level 4, since $\{x_5\} \cup \{x_6, x_7\}$ becomes complete in $G(4)$. At level 6, x_4, x_5, x_6, and x_7 lie for the first time in the same cluster. Although this subgraph is not complete, it does satisfy $h(2)$. Finally, at level 9, x_1, x_2, and x_3 come into the same cluster. Note that, although all nodes in the graph have node degree equal to 2, the final clustering will be formed at level 10 because at level 9 the graph is not connected.

Assume now that $h(k)$ is the node connectivity property, with $k = 2$; that is, all pairs of nodes in a connected subgraph are joined by at least two paths having no nodes in common. The dissimilarity dendrogram produced in this case is shown in Figure 13.10b.

Finally, the dissimilarity dendrogram produced when the edge connectivity property with $k = 2$ is employed is shown in Figure 13.10c.

It is not difficult to see that all these properties for $k = 1$ result in the single link algorithm. On the other hand, as k increases, the resulting subgraphs approach completeness.

Example 13.6

Consider again the dissimilarity matrix of the previous example. Assume now that $h(k)$ is the node degree property with $k = 3$. The corresponding dendrogram is shown in Figure 13.11. Comparing the dendrograms of Figures 13.10a and 13.11, we observe that the same clusters in the second case are formed in larger dissimilarity levels.

Clustering Algorithms Based on the Minimum Spanning Tree

A *spanning tree* is a connected graph (containing all the vertices of the graph) and having no loops (that is, there exists only one path connecting any two pairs of

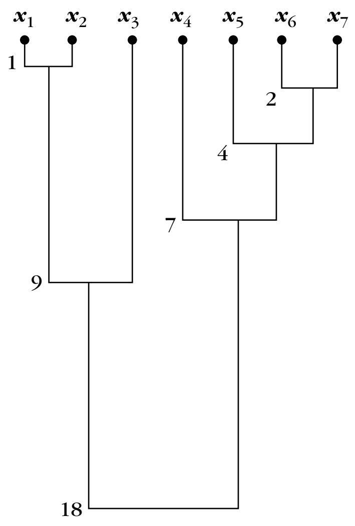


FIGURE 13.11

Threshold dendrogram related to Example 13.6 for the node degree property and $k = 3$.

nodes in the graph). If the edges of the graph are weighted, we define as the *weight of the spanning tree* the sum of the weights of its edges. A *minimum spanning tree (MST)* is a spanning tree with the smallest weight among all spanning trees connecting the nodes of the graph. An MST of a graph may be derived with Prim's algorithm or Kruskal's algorithm (e.g., see [Horo 78]). Note that there may be more than one minimum spanning trees for a given graph. However, when the weights of the edges of G are different from each other, then the MST is unique. In our case, the weights of a graph are derived from the proximity matrix $P(X)$.

Searching for the MST can also be seen as a special case of the GTAS scheme, if we adopt in place of $g_{h(k)}(C_r, C_s)$ the following proximity function:

$$g(C_r, C_s) = \min_{ij}\{w_{ij} : \boldsymbol{x}_i \in C_r,\ \boldsymbol{x}_j \in C_s\} \tag{13.26}$$

where $w_{ij} = d(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

In words, this measure identifies the minimum weight of the MST that connects the subgraphs corresponding to C_r and C_s.

Once the MST has been determined (using any suitable algorithm), we may identify a hierarchy of clusterings as follows: the clusters of the clustering at level t are identified as the *connected* components of G if only the edges of its MST with the smallest t weights are considered. It takes a little thought to see that this hierarchy is identical to the one defined by the single link algorithm, at least for the case in which all the distances between any two vectors of X are different from each other. Thus, this scheme may also be viewed as an alternative implementation of the single link algorithm. The following example demonstrates the operation of this scheme.

Example 13.7

Let us consider the following proximity matrix.

$$P = \begin{bmatrix} 0 & 1.2 & 4.0 & 4.6 & 5.1 \\ 1.2 & 0 & 3.5 & 4.2 & 4.7 \\ 4.0 & 3.5 & 0 & 2.2 & 2.8 \\ 4.6 & 4.2 & 2.2 & 0 & 1.6 \\ 5.1 & 4.7 & 2.8 & 1.6 & 0 \end{bmatrix}$$

The MST derived from this proximity matrix is given in Figure 13.12a. The corresponding dendrogram is given in Figure 13.12b.

It is easy to observe that a minimum spanning tree uniquely specifies the dendrogram of the single link algorithm. Thus, MST can be used as an alternative to the single link algorithm and can lead to computational savings.

13.2.6 Ties in the Proximity Matrix

In cases in which the vectors consist of interval-scaled or ratio-scaled features, the probability of a vector of the data set X being equidistant from two other vectors of X is very small for most practical problems. However, if we deal with ordinal data,

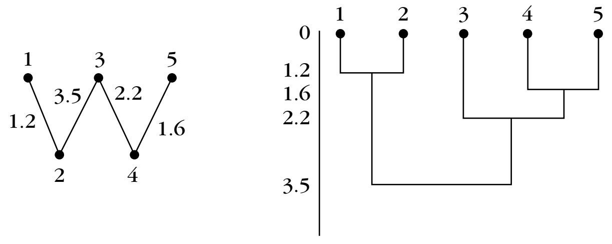


FIGURE 13.12

(a) The minimum spanning tree derived with the dissimilarity matrix given in Example 13.7. (b) The dissimilarity dendrogram obtained with the algorithm based on the MST.

this probability is not negligible. The fact that a vector is equidistant from two other vectors implies that a proximity matrix P will have at least two equal entries in the triangle above its main diagonal (see Example 13.8). It is interesting to see how the hierarchical algorithms behave with such proximity matrices. Let us consider first the family of algorithms based on the graph theory via the following example.

Example 13.8

Consider the following dissimilarity matrix:

$$P = \begin{bmatrix} 0 & 4 & 9 & 6 & 5 \\ 4 & 0 & 3 & 8 & 7 \\ 9 & 3 & 0 & 3 & 2 \\ 6 & 8 & 3 & 0 & 1 \\ 5 & 7 & 2 & 1 & 0 \end{bmatrix}$$

Note that $P(2,3) = P(3,4)$. The corresponding dissimilarity graph $G(9)$ is shown in Figure 13.13a. Figure 13.13b shows the corresponding dissimilarity dendrogram obtained by the single link algorithm. No matter which of the two edges is considered first, the resulting dendrogram remains the same. Figure 13.13c (13.13d) depicts the dendrogram obtained by the complete link algorithm when the $(3,4)$ ($(2,3)$) edge is considered first. Note that the dendrograms of Figures 13.13c and 13.13d are different.

Let us interchange the $P(1,2)$ and $P(2,3)$ entries of P[6] and let P_1 be the new dissimilarity matrix. Figure 13.14a shows the dendrogram obtained by the single link algorithm, and Figure 13.14b depicts the dendrogram obtained by the complete link algorithm. In this case, the complete link algorithm produces the same dendrogram regardless the order in which edges $(1,2)$ and $(3,4)$ are considered.

This example indicates that the single link algorithm leads to the same dendrogram, regardless of how the ties are considered. On the other hand, the complete

[6] Since a dissimilarity matrix is symmetric, $P(2,1)$ and $P(3,2)$ entries are also interchanged.

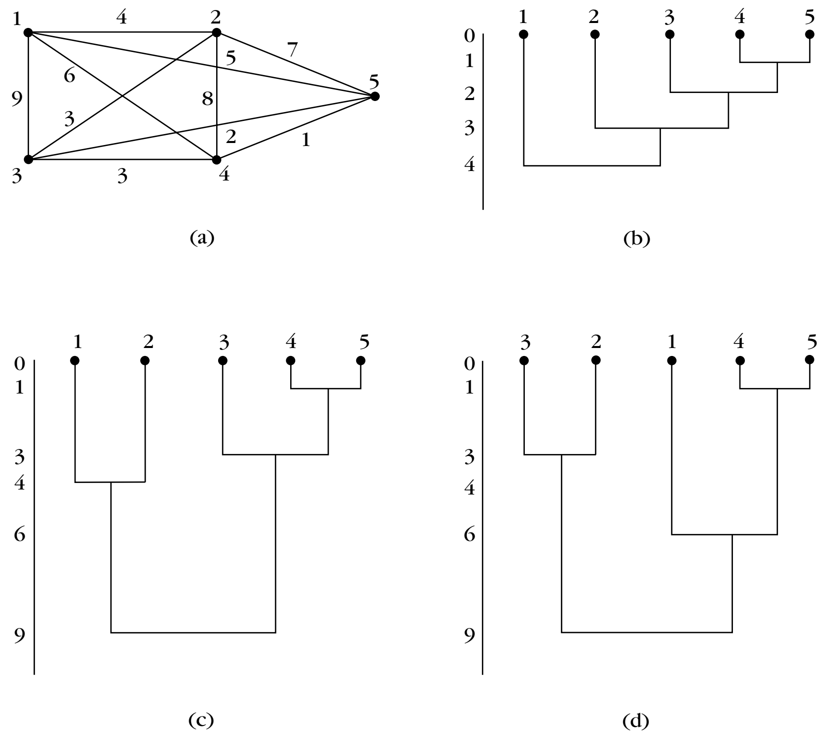
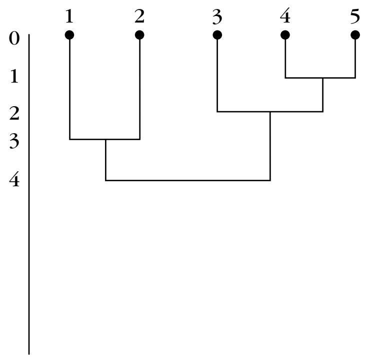


FIGURE 13.13

(a) The dissimilarity graph ($G(9)$) for the dissimilarity matrix given in Example 13.8. (b) The dissimilarity dendrogram obtained by the single link algorithm. (c) The dissimilarity dendrogram obtained by the complete link algorithm when edge $(3, 4)$ is considered first. (d) The dissimilarity dendrogram obtained by the complete link algorithm when edge $(2, 3)$ is considered first.

link algorithm *may* lead to different dendrograms if it follows different ways of considering the ties. The other graph theory–based algorithms, which fall between the single and the complete algorithm, exhibit behavior similar to that of the complete link algorithm (see Problem 13.11).

The same trend is true for the matrix-based algorithms. *Note, however, that in matrix-based schemes ties may appear at a later stage in the proximity matrix* (see Problem 13.12). Thus, as is shown in [Jard 71], the single link algorithm treats the ties in the proximity matrix in the most reliable way; it always leads to the same proximity dendrogram. It seems that every requirement additional to the connectivity property (for graph theory-based algorithms) or to Eq. (13.4) (for matrix theory-based algorithms) produces ambiguity, and the results become sensitive to the order in which ties are processed. From this point of view, the single link algorithm seems to outperform its competitors. This does not mean that all



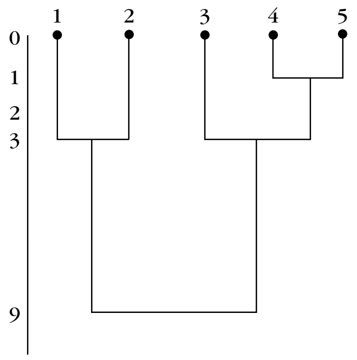


FIGURE 13.14

(a) The dissimilarity dendrogram obtained by the single link algorithm for $\boldsymbol{P}_1$ given in Example 13.8. (b) The dissimilarity dendrogram obtained by the complete link algorithm for $\boldsymbol{P}_1$ given in Example 13.8.

the other algorithms are inferior. The algorithm that gives the best results is problem dependent. However, if one decides to use any other algorithm, different from the single link, he or she must treat the possible ties in the proximity matrix very carefully.

13.3 THE COPHENETIC MATRIX

Another quantity associated with the hierarchical algorithms is the *cophenetic matrix*. This will be used as a tool for the validation of clustering hierarchies in Chapter 16.

Let $\Re_{t_{ij}}$ be the clustering at which $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are merged in the same cluster *for the first time* (of course, they will remain in the same cluster for all subsequent clusterings). Also let $L(t_{ij})$ be the proximity level at which clustering $\Re_{t_{ij}}$ is defined. We define the *cophenetic distance* between two vectors as

$$d_C(\boldsymbol{x}_i, \boldsymbol{x}_j) = L(t_{ij})$$

In words, the cophenetic distance between two vectors $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is defined as the proximity level at which the two vectors are found in the same cluster for the first time.

The cophenetic distance is a metric, *under the assumption of monotonicity*. To prove this, we need to show that the five conditions stated in Chapter 11 are satisfied. Indeed, the first condition holds; that is, $0 \leq d_C(\boldsymbol{x}, \boldsymbol{y}) < +\infty$. Note that the minimum value of d_C is zero, since $L(0) = 0$. Also, $d_C(\boldsymbol{x}, \boldsymbol{x}) = 0$, since a vector $\boldsymbol{x}$ lies in the same cluster with itself at the zero level clustering, where dissimilarity is 0. Furthermore, it is obvious that $d_C(\boldsymbol{x}, \boldsymbol{y}) = d_C(\boldsymbol{y}, \boldsymbol{x})$. Also, it is easy to see that $d_C(\boldsymbol{x}, \boldsymbol{y}) = 0 \Leftrightarrow \boldsymbol{x} = \boldsymbol{y}$, since at $L(0)$ each cluster consists of a single vector. Finally, the triangular inequality is also valid. Indeed, for a triple $(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_r)$, let $t_1 = \max\{t_{ij}, t_{jr}\}$ and $L_1 = \max\{L(t_{ij}), L(t_{jr})\}$. It is clear, from the property of hierarchy, that in the clustering formed at the t_1 level, $\boldsymbol{x}_i$, $\boldsymbol{x}_j$, and $\boldsymbol{x}_r$ fall into the same cluster. Furthermore, *assuming monotonicity*, $d_C(\boldsymbol{x}_i, \boldsymbol{x}_r) \leq L_1$, or

$$d_C(\boldsymbol{x}_i, \boldsymbol{x}_r) \leq \max\{d_C(\boldsymbol{x}_i, \boldsymbol{x}_j), d_C(\boldsymbol{x}_j, \boldsymbol{x}_r)\} \tag{13.27}$$

Note that two of these three distances are always equal, depending on which pair of vectors came first under a single cluster. This condition is stronger than the triangular inequality and is called *ultrametric inequality*. Notice the close relation of ultrametricity and monotonicity. Monotonicity ensures ultrametricity and, as a consequence, the triangular inequality.

The *cophenetic matrix* is defined as

$$D_C(X) = [d_C(\boldsymbol{x}_i,\ \boldsymbol{x}_j)] = [L(t_{ij})], \quad i,j = 1, \ldots, N$$

It is clear that the cophenetic matrix is symmetric. Moreover, apart from its diagonal elements, it has only $N-1$ distinct entries; that is, it has many ties (duplicate entries). $D_C(X)$ is a special case of dissimilarity matrix, since $d_C(\boldsymbol{x}_i, \boldsymbol{x}_j)$ satisfy the ultrametric inequality.

A hierarchical algorithm can, thus, be viewed as a mapping of the data proximity matrix into a cophenetic matrix.

Example 13.9

Let us consider the dissimilarity dendrogram of Figure 13.2b. The corresponding cophenetic matrix is

$$D_C(X) = \begin{bmatrix} 0 & 1 & 4.2 & 4.2 & 4.2 \\ 1 & 0 & 4.2 & 4.2 & 4.2 \\ 4.2 & 4.2 & 0 & 1.4 & 1.4 \\ 4.2 & 4.2 & 1.4 & 0 & 1.1 \\ 4.2 & 4.2 & 1.4 & 1.1 & 0 \end{bmatrix}$$

13.4 DIVISIVE ALGORITHMS

The divisive algorithms follow the reverse strategy from that of the agglomerative schemes. The first clustering contains a single set, X. At the first step, we search for the best possible partition of X into two clusters. The straightforward method is to consider all possible $2^{N-1}-1$ partitions of X into two sets and to select the optimum, according to a prespecified criterion. This procedure is then applied iteratively to each of the two sets produced in the previous stage. The final clustering consists of N clusters, each containing a single vector of X.

Let us state the general divisive scheme more formally. Here, the tth clustering contains $t+1$ clusters. In the sequel, C_{tj} will denote the jth cluster of the tth clustering $\Re_t$, $t=0,\ldots,N-1$, $j=1,\ldots,t+1$. Let $g(C_i, C_j)$ be a dissimilarity function defined for all possible pairs of clusters. The initial clustering $\Re_0$ contains only the set X, that is, $C_{01}=X$. To determine the next clustering, we consider all possible pairs of clusters that form a partition of X. Among them we choose the pair, denoted by (C_{11}, C_{12}), that maximizes g.[7] These clusters form the next clustering $\Re_1$, that is, $\Re_1=\{C_{11}, C_{12}\}$. At the next time step, we consider all possible pairs of clusters produced by C_{11} and we choose the one that maximizes g. The same procedure is repeated for C_{12}. Assume now that from the two resulting pairs of clusters, the one originating from C_{11} gives the larger value of g. Let this pair be denoted by (C_{11}^1, C_{11}^2). Then the new clustering, $\Re_2$, consists of C_{11}^1, C_{11}^2, and C_{12}. Relabeling these clusters as C_{21}, C_{22}, C_{23}, respectively, we have $\Re_2=\{C_{21}, C_{22}, C_{23}\}$. Carrying on in the same way, we form all subsequent clusterings. The general divisive scheme may be stated as follows:

Generalized Divisive Scheme (GDS)

```
■ Initialization
    • Choose $\Re_0 = \{X\}$ as the initial clustering.
    • $t = 0$
■ Repeat
    • $t = t + 1$
    • For $i = 1$ to $t$
        ○ Among all possible pairs of clusters $(C_r, C_s)$ that form a partition of $C_{t-1,i}$,
          find the pair $(C_{t-1,i}^1, C_{t-1,i}^2)$ that gives the maximum value for $g$.
    • Next $i$
    • From the $t$ pairs defined in the previous step choose the one that maximizes
      $g$. Suppose that this is $(C_{t-1,j}^1, C_{t-1,j}^2)$.
    • The new clustering is
          $$\Re_t = (\Re_{t-1} - \{C_{t-1,j}\}) \cup \{C_{t-1,j}^1, C_{t-1,j}^2\}$$
    • Relabel the clusters of $\Re_t$.
■ Until each vector lies in a single distinct cluster.
```

[7] We can also use a similarity function. In that case we should choose the pair of clusters that minimizes g.

Different choices of g give rise to different algorithms. One can easily observe that this divisive scheme is computationally very demanding, even for moderate values of N. This is its main drawback, compared with the agglomerative scheme. Thus, if these schemes are to be of any use in practice, some further computational simplifications are required. One possibility is to make compromises and not search for all possible partitions of a cluster. This can be done by ruling out many partitions as not "reasonable," under a prespecified criterion. Examples of such algorithms are discussed in [Gowd 95] and [MacN 64]. The latter scheme is discussed next.

Let C_i be an already formed cluster. Our goal is to split it further, so that the two resulting clusters, C_i^1 and C_i^2, are as "dissimilar" as possible. Initially, we have $C_i^1 = \emptyset$ and $C_i^2 = C_i$. Then, we identify the vector in C_i^2 whose average dissimilarity from the remaining vectors is maximum, and we move it to C_i^1. In the sequel, for each of the remaining $\boldsymbol{x} \in C_i^2$, we compute its average dissimilarity with the vectors of C_i^1, $g(\boldsymbol{x}, C_i^1)$, as well as its average dissimilarity with the rest of the vectors in C_i^2, $g(\boldsymbol{x}, C_i^2 - \{\boldsymbol{x}\})$. If for *every* $\boldsymbol{x} \in C_i^2$, $g(\boldsymbol{x}, C_i^2 - \{\boldsymbol{x}\}) < g(\boldsymbol{x}, C_i^1)$, then we stop. Otherwise, we select the vector $\boldsymbol{x} \in C_i^2$ for which the difference $D(\boldsymbol{x}) = g(\boldsymbol{x}, C_i^2 - \{\boldsymbol{x}\}) - g(\boldsymbol{x}, C_i^1)$ is maximum (among the vectors of C_i^2, $\boldsymbol{x}$ exhibits the maximum dissimilarity with $C_i^2 - \{\boldsymbol{x}\}$ and the maximum similarity to C_i^1) and we move it to C_i^1. The procedure is repeated until the termination criterion is met. This iterative procedure accounts for step 2.2.1 of the "generalized divisive scheme" GDS.

In the preceding algorithm the splitting of the clusters is based on all the features (coordinates) of the feature vectors. Algorithms of this kind are also called *polythetic algorithms*. In fact, all the algorithms that have been or will be considered in this book are polythetic. In contrast, there are divisive algorithms that achieve the division of a cluster based on a single feature at each step. These are the so-called *monothetic algorithms*. Such algorithms are discussed in [Ever 01]. For more details see [Lamb 62, Lamb 66, MacN 65].

A large research effort has been devoted to comparing the performance of a number of the various hierarchical algorithms in the context of different applications. The interested reader may consult, for example, [Bake 74, Hube 74, Kuip 75, Dube 76, Mill 80, Mill 83].

13.5 HIERARCHICAL ALGORITHMS FOR LARGE DATA SETS

As we have seen in Section 13.2 the number of operations for the generalized agglomerative scheme (GAS) is of the order of N^3, and this cannot become less than $O(N^2)$, even if efficient computational schemes are employed. This section is devoted to a special type of hierarchical algorithms that are most appropriate for handling large data sets. As it has been stated elsewhere, the need for such algorithms stems from a number of applications, such as Web mining, bioinformatics, and so on.

The CURE Algorithm

The acronym CURE stands for Clustering Using REpresentatives. The innovative feature of CURE is that it represents each cluster, C, by a set of $k > 1$ representatives, denoted by R_C. By using multiple representatives for each cluster, the CURE algorithm tries to "capture" the shape of each one. However, in order to avoid taking into account irregularities in the border of the cluster, the initially chosen representatives are "pushed" toward the mean of the cluster. This action is also known as "shrinking" in the sense that the volume of space "defined" by the representatives is shrunk toward the mean of the cluster. More specifically, for each C the set R_C is determined as follows:

- Select the point $\boldsymbol{x} \in C$ with the maximum distance from the mean of C and set $R_C = \{\boldsymbol{x}\}$ (the set of representatives).
- For $i = 2$ to $\min\{k, n_C\}$[8]
  - Determine $\boldsymbol{y} \in C - R_C$ that lies farthest from the points in R_C and set $R_C = R_C \cup \{\boldsymbol{y}\}$.
- End { For }
- Shrink the points $\boldsymbol{x} \in R_C$ toward the mean $\boldsymbol{m_c}$ in C by a factor a. That is, $\boldsymbol{x} = (1 - a)\boldsymbol{x} + a\boldsymbol{m_c}, \forall\ \boldsymbol{x} \in R_C$.

The resulting set R_C is the set of representatives of C. The distance between two clusters C_i and C_j is defined as

$$d(C_i, C_j) = \min_{\boldsymbol{x} \in R_{C_i},\, \boldsymbol{y} \in R_{C_j}} d(\boldsymbol{x}, \boldsymbol{y}) \tag{13.28}$$

Given the previous definitions, the CURE algorithm may be viewed as a special case of the *GAS* scheme. In its original version, the representatives of a cluster, C_q, generated from the agglomeration of two clusters C_i and C_j, are determined by taking into account all the points of C_q and applying the procedure described previously. However, to reduce the time complexity of this procedure, especially for the case of large number of points in each cluster, the representatives of C_q are selected among the $2k$ representatives of C_i and C_j. Such a choice is justified by the fact that the representatives are the most scattered points in each one of the clusters, C_i and C_j. Hence, it is expected that the resulting k representative points for C_q will also be well scattered. Once the final number of m clusters has been established, each point $\boldsymbol{x}$ in X, which is not among the representatives of any of the final clusters, is assigned to the cluster that contains the closest to $\boldsymbol{x}$ representative. The number m of clusters is either supplied to the algorithm by the user, based on his prior knowledge about the structure of the data set, or estimated using methods such as those described in Section 13.6.

[8] n_C denotes the number of points in C.

The worst-case time complexity of CURE can be shown to be $O(N^2 \log_2 N)$ ([Guha 98]). Clearly, this time complexity becomes prohibitive in the case of very large data sets. The technique adopted by the CURE algorithm, in order to reduce the computational complexity, is that of *random sampling*. That is, a sample set X' is created from X, by choosing *randomly* N' out of the N points of X. However, one has to ensure that the probability of missing a cluster of X, due to this sampling, is low. This can be guaranteed if the number of points N' is sufficiently large ([Guha 98]).

Having estimated N', CURE forms a number of $p = N/N'$ sample data sets by successive random sampling. In other words, X is partitioned randomly in p subsets. Let $q > 1$ be a user-defined parameter. Then, the points in each partition are clustered (following the procedure already explained) until N'/q clusters are formed or the distance between the closest pair of clusters to be merged in the next iteration step exceeds a user-defined threshold. Once the clustering procedure applied to each one of the subsets is completed, a second clustering pass on the (at most) $p(N'/q) = N/q$ clusters, obtained from all subsets, is performed. The goal of this pass is to apply the merging procedure described previously to all (at most) N/q clusters so that we end up with the required final number, m, of clusters. To this end, for each of the (at most) N/q clusters k representatives are used. Finally, each point $\boldsymbol{x}$ in the data set, X, that is not used as a representative in any one of the m clusters is assigned to one of them according to the following strategy. First, a random sample of representative points from each of the m clusters is chosen. Then, based on the previous representatives the point $\boldsymbol{x}$ is assigned to the cluster that contains the representative closest to it.

Experiments reported in [Guha 98] show that CURE is sensitive to the choice of the parameters k, N', and a. Specifically, k must be large enough to capture the geometry of each cluster. In addition, N' must be higher than a certain percentage of N (for the reported experiments, N' should be at least 2.5% of N). The choice of the shrinking factor, a, also affects the performance of CURE. For small values of a (small shrinkage), CURE exhibits a behavior similar to the MST algorithm, whereas for large values of a its performance resembles that of algorithms using a single point representative for each cluster. The worst-case execution time for CURE increases quadratically with the sample size N', that is, $O(N'^2 \log_2 N')$ ([Guha 98]).

Remarks

- The algorithm exhibits low sensitivity with respect to outliers within the clusters. The reason is that shrinking the scattered points toward the mean "dampens" the adverse effects due to outliers ([Guha 98]).
- The problem of outliers that form clusters by themselves is faced by CURE as follows. Because the outliers usually form small clusters, a few stages prior to the termination of the algorithm a check for clusters containing very few points takes place. These clusters are likely to consist of outliers, and they are removed.

- If N'/q is chosen to be sufficiently large compared to the final number (m) of clusters in X, it is expected that the points in a subset X' that are merged to the same cluster during the application of CURE on X', will also belong to the same cluster (as if the entire data set, X, were taken into account). In other words, it is expected that the quality of the final clustering obtained by CURE will not be significantly affected by the partitioning of X.
- For $a = 1$, all representatives diminish to the mean of the cluster.
- The CURE algorithm can reveal clusters with nonspherical or elongated shapes, as well as clusters of wide variance in size.
- In [Guha 98], the task of the efficient implementation of the algorithm is discussed using the heap and the $k - d$ tree data structures (see also [Corm 90] and [Same 89]).

The ROCK Algorithm

The RObust Clustering using linKs (ROCK) algorithm is best suited for nominal (categorical) features. For this type of data, it makes no sense to choose the mean of a cluster as a representative. In addition, the proximity between two feature vectors whose coordinates stem from a discrete data set cannot be adequately quantified by any of the l_p distances. ROCK introduces the idea of *links*, in place of distances, to merge clusters.

Before we proceed further, some definitions are in order. Two points $\boldsymbol{x}, \boldsymbol{y} \in X$ are considered *neighbors* if $s(\boldsymbol{x}, \boldsymbol{y}) \geq \theta$, where $s(\cdot)$ is an appropriately chosen *similarity* function and θ is a user-defined parameter, which defines the similarity level according to which two points can be considered "similar." Let $link(\boldsymbol{x}, \boldsymbol{y})$ be the number of *common* neighbors between $\boldsymbol{x}$ and $\boldsymbol{y}$. Consider the graph whose vertices are the points in X and whose edges connect points that are neighbors. Then, it can easily be determined that $link(\boldsymbol{x}, \boldsymbol{y})$ may be viewed as the number of distinct paths of length 2 connecting $\boldsymbol{x}$ and $\boldsymbol{y}$.

Assume now that there exists a function $f(\theta) < 1$ that is dependent on the data set as well as on the type of clusters we are interested in, with the following property: each point assigned to a cluster C_i has approximately $n_i^{f(\theta)}$ neighbors in C_i, where n_i is the number of points in C_i. Assuming that cluster C_i is large enough and that points outside C_i result in a very small number of links to the points of C_i, each point in C_i contributes approximately to $n_i^{2f(\theta)}$ *links*. Thus, the expected total number of *links* among all pairs in C_i is $n_i^{1+2f(\theta)}$ (Problem 13.15).

The "closeness" between two clusters is assessed by the function

$$g(C_i, C_j) = \frac{link(C_i, C_j)}{(n_i + n_j)^{1+2f(\theta)} - n_i^{1+2f(\theta)} - n_j^{1+2f(\theta)}} \tag{13.29}$$

where $link(C_i, C_j) = \sum_{\boldsymbol{x} \in C_i, \boldsymbol{y} \in C_j} link(\boldsymbol{x}, \boldsymbol{y})$. Note that the denominator in this fraction is the expected total number of links *between* the two clusters. Clearly, the larger the value of $g(\cdot)$ the more similar the clusters C_i and C_j. The number of clusters, m, is a user-defined parameter. At each iteration, the clusters with maximum $g(\cdot)$ are merged. The procedure stops when the number of clusters formed becomes equal to the desired number of clusters, m, or the number of *links* between every pair of clusters in a clustering $\Re_t$ is 0.

In the case where the data set X is very large (and in order to improve the execution time), ROCK is applied on a reduced data set, X', stemming from X via random sampling, whose size N' is estimated as in the CURE algorithm. After the clusters in the sample data set X' have been identified, the points in X that were not selected in the sample subset are assigned to a cluster via the following strategy. First, a subset L_i of points from each cluster C_i is selected. In the sequel, for each $\boldsymbol{z}$ in $X - X'$ and each cluster C_i, the number N_i of its neighbors among the L_i points is determined. Then $\boldsymbol{z}$ is assigned to the cluster C_i for which the quantity $N_i/(n_{L_i} + 1)^{f(\theta)}$ is maximum, where n_{L_i} is the number of points in L_i. Note that the denominator in this expression is the expected number of neighbors of $\boldsymbol{z}$ in $L_i \cup \{\boldsymbol{z}\}$.

Remarks

- In [Guha 00], it is proposed using $f(\theta) = (1 - \theta)/(1 + \theta)$, with $\theta < 1$.
- The algorithm makes a rather strong hypothesis about the existence of the function $f(\theta)$. In other words, it poses a constraint for each cluster in the data set, which may lead to poor results if the clusters in the data set do not satisfy this hypothesis. However, in the experimental cases discussed in [Guha 00] the clustering results were satisfactory for the choice of $f(\theta)$.
- For large values of N, the worst-case time complexity of ROCK is similar to that of CURE.

In [Dutt 05] it is proved that, under certain conditions, the clusters produced by the ROCK algorithm are the connected components of a certain graph, called *link graph* whose vertices correspond to the data points. Based on the above result a quick version of ROCK, called QROCK, is described, which simply identifies the connected components of the link graph.

The Chameleon Algorithm

The algorithms CURE and ROCK, described previously, are based on "static" modeling of the clusters. Specifically, CURE models each cluster by the same number, k, of representatives, whereas ROCK poses constraints on the clusters through $f(\theta)$. Clearly, these algorithms may fail to unravel the clustering structure of the data set in cases where the individual clusters do not obey the adopted model, or when noise is present. In the sequel, another hierarchical clustering algorithm, known as *Chameleon*, is presented. The algorithm is capable of recovering clusters of various shapes and sizes.

To quantify the similarity between two clusters, we define the concepts of *relative interconnectivity* and *relative closeness*. Both of these quantities are defined in terms of graph theory concepts. Specifically, a graph $G = (V, E)$ is constructed so that each data point corresponds to a vertex in V and E contains edges among vertices. The weight of each edge is set equal to the *similarity* between the points associated with the connected vertices.

Before we proceed further, the following definitions are in order. Let C be the set of points corresponding to a subset of V. Assume that C is partitioned into two nonempty subsets C_1 and C_2 ($C_1 \cup C_2 = C$). The subset of the edges $E^{'}$ of E that connect C_1 and C_2 form the *edge cut set*. If the sum of the weights of the edge cut set, $E^{'}$, corresponding to the (C_1, C_2) partition of C is minimum among all edge cut sets resulting from all possible partitions of C into two sets (excluding the empty set cases), $E^{'}$ is called the *minimum cut set* of C. If, now, C_1 and C_2 are constrained to be of approximately equal size, then the minimum sum $E^{'}$ (over all possible partitions of approximately equal size) is known as the *minimum cut bisector* of C. For example, suppose that C is represented by the vertices v_1, v_2, v_3, v_4, and v_5 in Figure 13.15. The edges of the respective graph G are e_{12}, e_{23}, e_{34}, e_{45}, and e_{51}, with weights 0.1, 0.2, 0.3, 0.4, and 0.5, respectively. Then, the sums of the weights of the edges of the sets $\{e_{51}, e_{34}\}$, $\{e_{12}, e_{23}\}$, and $\{e_{12}, e_{34}\}$ are 0.8, 0.3, and 0.4, respectively. The second edge cut set corresponds to the minimum cut set of C, whereas the third corresponds to the minimum cut bisector of C.

Relative Interconnectivity

Let E_{ij} be the set of edges connecting points in C_i with points in C_j, and E_i be the set of edges that corresponds to the minimum cut bisector of C_i. The *absolute interconnectivity*, $|E_{ij}|$, between two clusters C_i and C_j, is defined as the sum of the weights of the edges in E_{ij}. Equivalently, this is the edge cut set associated with the partition of $C_i \cup C_j$ into C_i and C_j. The *internal interconnectivity* $|E_i|$ of a cluster C_i is defined as the sum of the weights of its minimum cut bisector E_i. The relative

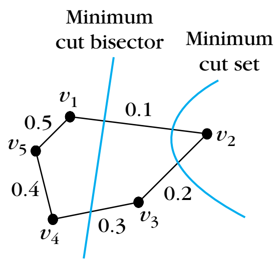


FIGURE 13.15

Partitions leading to the minimum cut set and the minimum cut bisector of the set C containing five vertices (see text for explanation).

interconnectivity between two clusters, C_i and C_j, is defined as

$$RI_{ij} = \frac{|E_{ij}|}{\frac{|E_i|+|E_j|}{2}} \tag{13.30}$$

Relative Closeness

Let S_{ij} be the average weight of the edges in the set E_{ij}, and S_i be the average weight of the edges in E_i. Then the relative closeness between two clusters C_i and C_j is defined as

$$RC_{ij} = \frac{S_{ij}}{\frac{n_i}{n_i+n_j}S_i + \frac{n_j}{n_i+n_j}S_j} \tag{13.31}$$

where n_i and n_j are the number of points in C_i and C_j, respectively.

Having defined the previous quantities, we proceed now to description of the algorithm. This algorithm, unlike most hierarchical algorithms that are either agglomerative or divisive, enjoys both of these characteristics. Initially, a k-nearest neighbor graph is created. More specifically, each vertex of the graph corresponds to a feature vector, and an edge is added between two vertices if *at least* one of the corresponding points is among the k-nearest neighbors of the other (typically k takes values in the range from 5 to 20). Note that if a point $\boldsymbol{x}$ is among the k nearest neighbors of a point $\boldsymbol{y}$, this does not necessarily mean that $\boldsymbol{y}$ is among the k nearest neighbors of $\boldsymbol{x}$.

The first phase of the algorithm is the divisive phase. Initially, all points belong to a single cluster. This cluster is partitioned into two clusters so that the sum of the weights of the edge cut set between the resulting clusters is minimized and each of the resulting clusters contains at least 25% of the vertices of the initial cluster. Then, at each step the largest cluster is selected and is partitioned as indicated previously. This procedure terminates when all obtained clusters, at a given level, contain fewer than q points (a user-defined parameter, typically chosen in the range 1 to 5% of the total number of points in X). In the sequel, the agglomerative phase is applied on the set of clusters that have resulted from the previous phase. Specifically, two clusters C_i and C_j are merged to a single cluster if

$$RI_{ij} \geq T_{RI} \quad \text{and} \quad RC_{ij} \geq T_{RC} \tag{13.32}$$

where T_{RI} and T_{RC} are user-defined parameters. Observe that for merging two clusters their internal structure plays an important role through their respective S and internal interconnectivity values, as (13.30) and (13.31) suggest. The more similar the elements within each cluster the higher "their resistance" in merging with another cluster. If more than one clusters C_j satisfy both conditions for a given cluster C_i, then C_i is merged with the cluster for which $|E_{ij}|$ is the highest among the candidates C_js. In addition, unlike most agglomerative algorithms, it is permissible to merge more than one pair of clusters at a given level of hierarchy (provided, of course, that these pairs satisfy the condition (13.32)).

A different rule that may be employed for the choice of the clusters to be merged is the following: merge those clusters that maximize the quantity

$$RI_{ij}RC_{ij}^{a} \tag{13.33}$$

where a gives the relative importance between RI and RC. Typically, a is chosen between 1.5 and 3.

Example 13.10

To gain a better understanding of the rationale behind the Chameleon algorithm, let's consider the overly-simplistic four-cluster data set depicted in Figure 13.16. We assume that $k = 2$. The similarities between the connected pairs of points for C_1, C_2, C_3, and C_4 are as shown in the figure (the weights between the connected pairs of points in C_3 and C_4 not shown in the figure are all equal to 0.9). In addition, the similarity between the closest points between clusters C_1 and C_2 is 0.4, whereas the similarity between the closest points between clusters C_3 and C_4 is 0.6. Note that although C_3 and C_4 lie closer to each other than C_1 and C_2, each exhibits a significantly higher degree of internal interconnectivity, compared to each of the C_1 and C_2.

For the previous clusters we have $|E_1| = 0.48$, $|E_2| = 0.48$, $|E_3| = 0.9 + 0.55 = 1.45$, $|E_4| = 1.45$, $|S_1| = 0.48$, $|S_2| = 0.48$, $|S_3| = 1.45/2 = 0.725$, and $|S_4| = 0.725$. In addition, $|E_{12}| = 0.4$, $|E_{34}| = 0.6$, $|S_{12}| = 0.4$, and $|S_{34}| = 0.6$. Taking into account the definitions of RI and RC, we have $RI_{12} = 0.833$, $RI_{34} = 0.414$ and $RC_{12} = 0.833$, $RC_{34} = 0.828$. We see that both RI and RC favor the merging of C_1 and C_2 against the merging of C_3 and C_4.

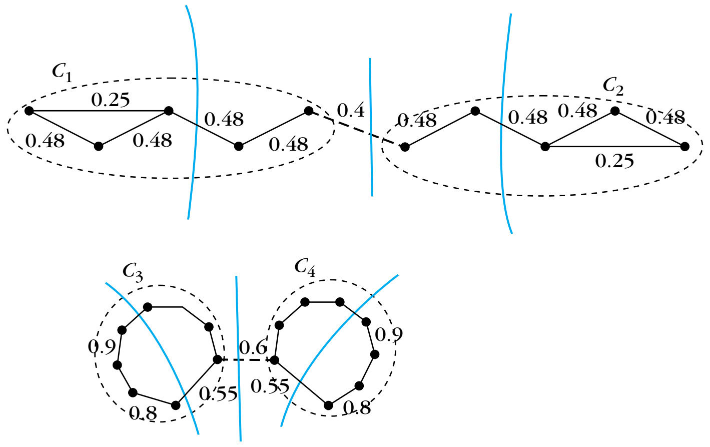


FIGURE 13.16

The high degree of internal interconnectivity of C_3 and C_4 favors the merging of C_1 with C_2, although the neighbors between C_1 and C_2 are less similar than the neighbors between C_3 and C_4.

Thus, Chameleon merges C_1 and C_2, which is in agreement with intuition. Note that the MST algorithm would merge the clusters C_3 and C_4.

Remarks

- This algorithm requires user-defined parameters (k, q and T_{RI}, T_{RC} for the rule given in (13.32) or a for the rule given in (13.33)). Experiments reported in [Kary 99] show that Chameleon is not very sensitive to the choice of the parameters k, q, and a.
- Chameleon requires large data sets in order to have more accurate estimates for $|E_{ij}|$, $|E_i|$, S_{ij}, and S_i. Thus, the algorithm is well suited for the applications where the volume of the available data is large.
- For large N, the worst-case time complexity of the algorithm is $O(N(\log_2 N + m))$, where m is the number of clusters formed after completion of the first (divisive) phase of the algorithm.

An alternative clustering method that employs agglomerative algorithms and is suitable for processing very large data sets is the so-called BIRCH method (Balanced Iterative Reducing and Clustering using Hierarchies) ([Zhan 96]). This method has been designed so as to minimize the number of I/O operations (that is, the exchange of information between the main memory and the secondary memory where the data set is stored). It performs a preclustering of data and stores a compact summary for each generated subcluster in a specific data structure called $CF - tree$. More specifically, it generates the maximum number of subclusters such that the resulting $CF - tree$ fits in the main memory. Then an agglomerative clustering algorithm ([Olso 93]) is applied on the subcluster summaries to produce the final clusters. BIRCH can achieve a computational complexity of $O(N)$. Two generalizations of BIRCH, known as BUBBLE and BUBBLE-FM algorithms, are given in [Gant 99].

13.6 CHOICE OF THE BEST NUMBER OF CLUSTERS

So far, we have focused on various hierarchical algorithms. In the sequel we turn our attention to the important task of determining the best clustering within a given hierarchy. Clearly, this is equivalent to identifying the number of clusters that best fits the data. An intuitive approach is to search in the proximity dendrogram for clusters that have a large *lifetime*. The lifetime of a cluster is defined as the absolute value of the difference between the proximity level at which it is created and the proximity level at which it is absorbed into a larger cluster. For example, the dendrogram of Figure 13.17a suggests that two major clusters are present and that of Figure 13.17b suggests only one. In [Ever 01], experiments are conducted to

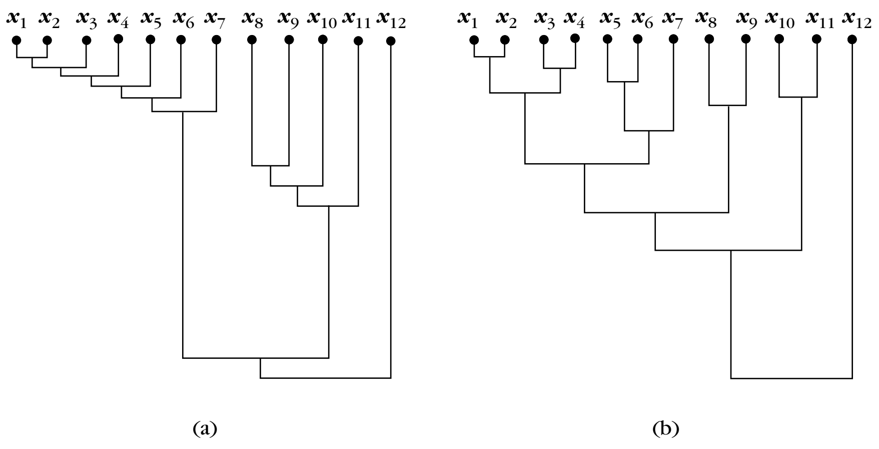

x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10} x_{11} x_{12}
(a)
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10} x_{11} x_{12}
(b)

FIGURE 13.17

(a) A dendrogram that suggests that there are two major clusters in the data set. (b) A dendrogram indicating that there is a single major cluster in the data set.

assess the behavior of various agglomerative algorithms when (a) a single compact cluster and (b) two compact clusters are formed by the vectors of X. However, human subjectivity is required to reach conclusions.

Many formal methods that may be used in cooperation with both hierarchical and nonhierarchical algorithms for identifying the best number of clusters for the data at hand have been proposed (e.g., [Cali 74, Duda 01, Hube 76]).

A comparison of many such methods is given in [Mill 85]. In the sequel, we discuss two methods, proposed in [Bobe 93] for identifying the clustering that best fits the data that are appropriate for agglomerative algorithms. The clustering algorithm does not necessarily produce the whole hierarchy of N clusterings, but it terminates when the clustering that best fits the data has been achieved, according to a criterion.

Method I

This is an extrinsic method, in the sense that it requires determination of the value of a specific parameter by the user. It involves the definition of a function $h(C)$ that measures the dissimilarity between the vectors of the same cluster C. That is, we can view it as a "self-similarity" measure. For example, $h(C)$ may be defined as

$$h_1(C) = \max\{d(\boldsymbol{x},\boldsymbol{y}), \boldsymbol{x},\boldsymbol{y} \in C\} \tag{13.34}$$

or

$$h_2(C) = \text{med}\{d(\boldsymbol{x},\boldsymbol{y}), \boldsymbol{x},\boldsymbol{y} \in C\} \tag{13.35}$$

(see Figure 13.18a).

When d is a metric distance, $h(C)$ may be defined as

$$h_3(C) = \frac{1}{2n_C}\sum_{\boldsymbol{x}\in C}\sum_{\boldsymbol{y}\in C} d(\boldsymbol{x},\boldsymbol{y}) \tag{13.36}$$

where n_C is the cardinality of C. Other definitions of $h(C)$ are also possible for the last case.

Let θ be an appropriate threshold for the adopted $h(C)$. Then, the algorithm terminates at the $\Re_t$ clustering if

$$\exists C_j \in \Re_{t+1} : h(C_j) > \theta \tag{13.37}$$

In words, $\Re_t$ is the final clustering if there exists a cluster C in $\Re_{t+1}$ with dissimilarity between its vectors ($h(C)$) greater than θ.

Sometimes the threshold θ is defined as

$$\theta = \mu + \lambda\sigma \tag{13.38}$$

where μ is the average distance between any two vectors in X and σ is its variance. The parameter λ is a user-defined parameter. Thus, the need for specifying an appropriate value for θ is transferred to the choice of λ. However, λ may be estimated more reasonably than θ.

Method II

This is an intrinsic method; that is, in this case only the structure of the data set X is taken into account. According to this method, the final clustering $\Re_t$ *must* satisfy the following relation:

$$d_{\min}^{ss}(C_i, C_j) > \max\{h(C_i), h(C_j)\}, \quad \forall C_i, C_j \in \Re_t \tag{13.39}$$

where $d_{\min}^{ss}$ is defined in Chapter 11. In words, in the final clustering, the dissimilarity between every pair of clusters is larger than the "self-similarity" of each of them (see Figure 13.18b). Note that this is only a necessary condition.

Finally, it must be stated that all these methods are based on heuristic arguments, and they are indicative only of the best clustering.

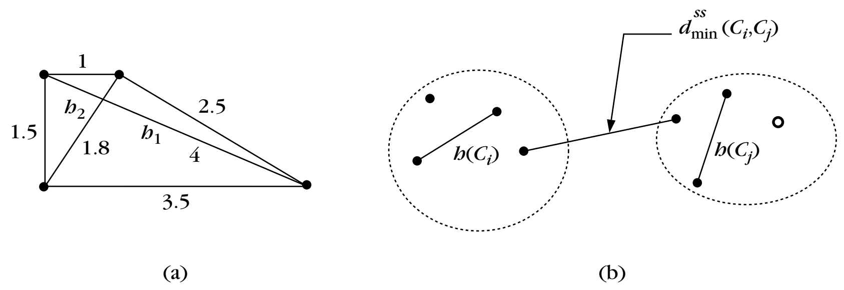


FIGURE 13.18

(a) Examples of "self-similarity" measures. (b) Illustration of the termination condition for method II.

13.7 PROBLEMS

13.1 Consider the Euclidean distance as the proximity measure between two vectors. Consider only ratio-scaled vectors. Prove that:

a. A pattern matrix uniquely identifies the corresponding proximity matrix.

b. A proximity matrix does not identify a pattern matrix uniquely. Furthermore, there are proximity matrices which do not correspond to any pattern matrix.

Hint: (b) Consider, for example, the translations of the points of the data set X.

13.2 Derive Eq. (13.10) from Eq. (13.8).
Hint: Make use of the following identities:

$$n_3\boldsymbol{m}_3 = n_1\boldsymbol{m}_1 + n_2\boldsymbol{m}_2 \tag{13.40}$$

and

$$n_1\|\boldsymbol{m}_1 - \boldsymbol{m}_3\|^2 + n_2\|\boldsymbol{m}_2 - \boldsymbol{m}_3\|^2 = \frac{n_1 n_2}{n_1 + n_2}\|\boldsymbol{m}_1 - \boldsymbol{m}_2\|^2 \tag{13.41}$$

where C_1 and C_2 are any two clusters and $C_3 = C_1 \cup C_2$.

13.3 Show that for the WPGMC algorithm there are cases where $d_{qs} \leq \min(d_{is}, d_{js})$.

13.4 Prove

$$d'_{qs} = \frac{n_q n_s}{n_q + n_s} d_{qs} \tag{13.42}$$

Hint: Multiply both sides of

$$\|\boldsymbol{m}_q - \boldsymbol{m}_s\|^2 = \frac{n_i}{n_i + n_j} d_{is} + \frac{n_j}{n_i + n_j} d_{js} - \frac{n_i n_j}{(n_i + n_j)^2} d_{ij}$$

by $(n_i + n_j)n_s/(n_i + n_j + n_s)$ (This equation holds from Problem 13.2).

13.5 **a.** Prove Eq. (13.16).

b. Complete the proof of Eq. (13.19).

Hint: Take the squares of both sides of Eq. (13.18).

13.6 Consider the proximity matrix given in Example 13.5. Find the proximity dendrograms derived by the GTAS algorithm when $h(k)$ is (a) the node connectivity property and (b) the edge connectivity property, with $k = 3$.

13.7 Prove that the distance between two clusters C_r and C_s, $d(C_r, C_s)$, which are at the same level of the hierarchy produced by the single link algorithm, given by Eq. (13.4), may be written as

$$d(C_r, C_s) = \min_{\boldsymbol{x} \in C_r, \boldsymbol{y} \in C_s} d(\boldsymbol{x}, \boldsymbol{y}) \tag{13.43}$$

That is, the single link algorithms based on matrix and graph theory are equivalent.
Hint: Proceed by induction on the level of hierarchy t. Take into account that the clusterings $\Re_t$ and $\Re_{t+1}$ have $N - t - 2$ common clusters.

13.8 Prove that the distance between two clusters C_r and C_s, $d(C_r, C_s)$, which are at the same level of the hierarchy produced by the complete link algorithm, given by Eq. (13.5), may be written as

$$d(C_r, C_s) = \max_{\boldsymbol{x} \in C_r, \boldsymbol{y} \in C_s} d(\boldsymbol{x}, \boldsymbol{y}) \tag{13.44}$$

That is, the complete link algorithms based on matrix and graph theory are equivalent.
Hint: Take into account the hints of the previous problem.

13.9 State and prove the propositions of the previous two problems when similarity measures between two vectors are used.

13.10 Consider the following proximity matrix:

$$P = \begin{bmatrix} 0 & 4 & 9 & 6 & 5 \\ 4 & 0 & 1 & 8 & 7 \\ 9 & 1 & 0 & 3 & 2 \\ 6 & 8 & 3 & 0 & 1 \\ 5 & 7 & 2 & 1 & 0 \end{bmatrix}$$

Apply the single and complete link algorithms to P and comment on the resulting dendrograms.

13.11 Consider the dissimilarity matrix P given in Example 13.5. Let us change $P(3, 4)$ to 6 ($P(4, 6)$ is also equal to 6). Also let $h(k)$ be the node degree property with $k = 2$. Run the corresponding graph theory algorithm when (a) the edge $(3, 4)$ is considered first and (b) the edge $(4, 6)$ is considered first. Comment on the resulting dendrograms.

13.12 Consider the following dissimilarity matrix:

$$P = \begin{bmatrix} 0 & 4 & 9 & 6 & 5 \\ 4 & 0 & 3 & 8 & 7 \\ 9 & 3 & 0 & 3 & 2 \\ 6 & 8 & 3 & 0 & 1 \\ 5 & 7 & 2 & 1 & 0 \end{bmatrix}$$

a. Determine all possible dendrograms resulting from application of the single and the complete link algorithms to P and comment on the results.

b. Set $P(3,4) = 4, P(1,2) = 10$, and let P_1 be the new proximity matrix. Note that P_1 contains no ties. Determine all possible dendrograms resulting from the application of the UPGMA algorithm to P_1.

13.13 Consider the general divisive scheme. Assume that at step 2.2.1 of the algorithm the cluster $C^1_{t-1,i}$ consists of a single vector, for $i = 1,\ldots,t$, $t = 1,\ldots,N$. Compute the number of pairs of clusters that have to be examined during the whole clustering process. Discuss the merits and the disadvantages of this scheme.

13.14 Does the alternative divisive algorithm discussed in Section 13.4 guarantee the determination of the optimum possible clustering at each level? Is it reasonable to extend the answer to this question to other such divisive algorithms?

13.15 In the ROCK algorithm prove that the expected total number of links among all pairs of points in a cluster, C, is $n^{1+2f(\theta)}$, where n is the cardinality of C.

13.16 Explain the physical meaning of Eq. (13.38).

MATLAB PROGRAMS AND EXERCISES

Computer Programs

13.1 *Convert matrix to vector.* Write a MATLAB function named *convert_prox_mat* that takes as input an $N \times N$ dimensional proximity matrix *prox_mat* and returns an $N(N-1)/2$ dimensional row vector *proc_vec* that contains the upper diagonal elements of *prox_mat* in the following order: $(1,2)$, $(1,3)$, $\ldots(1,N)$, $(2,3)$, $\ldots(2,N)$, $\ldots(N-1,N)$.

Solution

```
function prox_vec=convert_prox_mat(prox_mat)
  [N,N]=size(prox_mat);
  prox_vec=[];
  for i=1:N-1
    prox_vec=[prox_vec prox_mat(i,i+1:N)];
  end
```

13.2 *Single link, Complete link algorithms.* Write MATLAB code for the single link and complete link algorithms.

Solution

Just type

```
Z=linkage(prox_vec,'single')
```

for the single link algorithm and

```
Z=linkage(prox_vec,'complete')
```

for the complete link algorithm. The function *linkage* is a built-in MATLAB function that takes as inputs (a) a proximity vector in the form described before in 13.1 and (b) the type of the agglomerative algorithm that will be used. It returns an $(N-1) \times 3$ matrix Z, each line of which corresponds to a clustering of the hierarchy. The first two elements of the ith line of Z contain the indices of the objects that were linked at this level to form a new cluster. The index value $N+i$ is assigned to the new cluster. If, for example, $N=7$ and at the first level the elements 1 and 4 are merged, the cluster consisting of these elements will be represented by the integer $8(=7+1)$. If cluster 8 appears at a later row, this means that this cluster is being combined with another cluster in order to form a larger cluster. Finally, the third element of the ith line of Z contains the dissimilarity between the clusters that have been merged at this level.

13.3 Generate the dendrogram corresponding to the output of an agglomerative algorithm.

Solution

Just type

```
H=dendrogram(Z);
```

The *dendrogram* is a built-in MATLAB function that takes as input the output of the *linkage* function and generates the corresponding dendrogram.

Computer Experiments

13.1 Consider the following dissimilarity matrix

$$prox\_mat = \begin{bmatrix}
0.0 & 2.0 & 4.2 & 6.6 & 9.2 & 12.0 & 15.0 & 300 & 340 & 420 \\
2.0 & 0.0 & 2.2 & 4.6 & 7.2 & 10.0 & 13.0 & 280 & 320 & 400 \\
4.2 & 2.2 & 0.0 & 2.4 & 5.0 & 7.8 & 10.8 & 270 & 310 & 390 \\
6.6 & 4.6 & 2.4 & 0.0 & 2.6 & 5.4 & 8.4 & 260 & 300 & 380 \\
9.2 & 7.2 & 5.0 & 2.6 & 0.0 & 2.8 & 5.8 & 262 & 296 & 388 \\
12.0 & 10.0 & 7.8 & 5.4 & 2.8 & 0.0 & 3.0 & 316 & 280 & 414 \\
15.0 & 13.0 & 10.8 & 8.4 & 5.8 & 3.0 & 0.0 & 380 & 326 & 470 \\
300 & 280 & 270 & 260 & 262 & 316 & 380 & 0.0 & 4.0 & 4.4 \\
340 & 320 & 310 & 300 & 296 & 280 & 326 & 4.0 & 0.0 & 9.0 \\
420 & 400 & 390 & 380 & 388 & 414 & 470 & 4.4 & 9.0 & 0.0
\end{bmatrix}$$

a. Apply the single link and the complete link algorithms on *prox_mat* and draw the corresponding dissimilarity dendrograms.

b. Comment on the way the successive clusterings are created in each of the two algorithms.

c. Compare the dissimilarity thresholds where clusters are merged for both single link and complete link algorithms.

d. Can you draw any conclusion about the most natural clustering for the above data set by considering the dissimilarity thresholds where the clusters are merged for both algorithms?

REFERENCES

[Ande 73] Anderberg M.R. *Cluster Analysis for Applications*, Academic Press, 1973.

[Bake 74] Baker F.B. "Stability of two hierarchical grouping techniques. Case 1. Sensitivity to data errors," *J. Am. Statist. Assoc.*, Vol. 69, pp. 440–445, 1974.

[Bobe 93] Boberg J., Salakoski T. "General formulation and evaluation of agglomerative clustering methods with metric and non-metric distances," *Pattern Recognition*, Vol. 26(9), pp. 1395–1406, 1993.

[Cali 74] Calinski R.B., Harabasz J. "A dendrite method for cluster analysis," *Communications in Statistics*, Vol. 3, pp. 1–27, 1974.

[Corm 90] Cormen T.H., Leiserson C.E., Rivest R.L. *Introduction to Algorithms*, MIT Press, 1990.

[Day 84] Day W.H.E., Edelsbrunner H. "Efficient algorithms for agglomerative hierarchical clustering methods," *Journal of Classification*, Vol. 1(1), pp. 7–24, 1984.

[Dube 76] Dubes R., Jain A.K. "Clustering techniques: The user's dilemma," *Pattern Recognition*, Vol. 8, pp. 247–260, 1976.

[Dube 87] Dubes R. "How many clusters are best?—An experiment," *Pattern Recognition*, Vol. 20(6), pp. 645–663, 1987.

[Duda 01] Duda R., Hart P., Stork D. *Pattern Classification*, 2nd ed., John Wiley & Sons, 2001.

[Dutt 05] Dutta M., Mahanta A.K., Pujari A.K. "QROCK: A quick version of the ROCK algorithm for clustering catergorical data," *Pattern Recognition Letters*, Vol. 26, pp. 2364–2373, 2005.

[El-G 68] El-Gazzar A., Watson L., Williams W.T., Lance G. "The taxonomy of Salvia: A test of two radically different numerical methods," *J. Linn. Soc. (Bot.)*, Vol. 60, pp. 237–250, 1968.

[Ever 01] Everitt B., Landau S., Leese M. *Cluster Analysis*, Arnold, 2001.

[Frig 97] Frigui H., Krishnapuram R. "Clustering by competitive agglomeration," *Pattern Recognition*, Vol. 30(7), pp. 1109–1119, 1997.

[Gant 99] Ganti V., Ramakrishnan R., Gehrke J., Powell A., French J. "Clustering large datasets in arbitrary metric spaces," *Proceedings 15th International Conference on Data Engineering*, pp. 502–511, 1999.

[Gowd 95] Gowda Chidananda K., Ravi T.V. "Divisive clustering of symbolic objects using the concepts of both similarity and dissimilarity," *Pattern Recognition*, Vol. 28(8), pp. 1277–1282, 1995.

[Gowe 67] Gower J.C. "A comparison of some methods of cluster analysis," *Biometrics*, Vol. 23, pp. 623–628, 1967.

[Guha 98] Guha S., Rastogi R., Shim K. "CURE: An efficient clustering algorithm for large databases," *Proceedings of the ACM SIGMOD Conference on Management of Data*, pp. 73–84, 1998.

[Guha 00] Guha S., Rastogi R., Shim K. "ROCK: A robust clustering algorithm for categorical attributes," *Information Systems*, Vol. 25, No. 5, pp. 345–366, 2000.

[Hods 71] Hodson F.R. "Numerical typology and prehistoric archaeology," in *Mathematics in Archaeological and Historical Sciences* (Hodson F.R., Kendell D.G., Tautu P.A., eds.), University Press, Edinburgh, 1971.

[Horo 78] Horowitz E., Sahni S. *Fundamentals of Computer Algorithms*, Computer Science Press, 1978.

[Hube 74] Hubert L.J. "Approximate evaluation techniques for the single link and complete link hierarchical clustering procedures," *J. Am.Statist. Assoc.*, Vol. 69, pp. 698–704, 1974.

[Hube 76] Hubert L.J., Levin J.R. "A general statistical framework for assessing categorical clustering in free recall," *Psychological Bulletin*, Vol. 83, pp. 1072–1080, 1976.

[Jain 88] Jain A.K., Dubes R.C. *Algorithms for Clustering Data*, Prentice Hall, 1988.

[Jard 71] Jardine N., Sibson R. *Numerical Taxonomy*, John Wiley & Sons, 1971.

[Kank 96] Kankanhalli M.S., Mehtre B.M., Wu J.K. "Cluster-based color matching for image retrieval," *Pattern Recognition*, Vol. 29(4), pp. 701–708, 1996.

[Kary 99] Karypis G., Han E., Kumar V. "Chameleon: Hierarchical clustering using dynamic modeling," *IEEE Computer*, Vol. 32, No. 8, pp. 68–75, 1999.

[Kuip 75] Kuiper F.K., Fisher L. "A Monte Carlo comparison of six clustering procedures," *Biometrics*, Vol. 31, pp. 777–783, 1975.

[Kuri 91] Kurita T. "An efficient agglomerative clustering algorithm using a heap," *Pattern Recognition*, Vol. 24, pp. 205–209, 1991.

[Lamb 62] Lambert J.M., Williams W.T. "Multivariate methods in plant technology, IV. Nodal analysis," *J. Ecol.*, Vol. 50, pp. 775–802, 1962.

[Lamb 66] Lambert J.M., Williams W.T. "Multivariate methods in plant technology, IV. Comparison of information analysis and association analysis," *J. Ecol.*, Vol. 54, pp. 635–664, 1966.

[Lanc 67] Lance G.N., Williams W.T. "A general theory of classificatory sorting strategies: II. Clustering systems," *Computer Journal*, Vol. 10, pp. 271–277, 1967.

[Li 90] Li X. "Parallel algorithms for hierarchical clustering and cluster validity," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 12(11), pp. 1088–1092, 1990.

[Ling 72] Ling R.F. "On the theory and construction of k-clusters," *Computer Journal*, Vol. 15, pp. 326–332, 1972.

[Liu 68] Liu C.L. *Introduction to Combinatorial Mathematics*, McGraw-Hill, 1968.

[MacN 64] MacNaughton-Smith P., Williams W.T., Dale M.B., Mockett L.G. "Dissimilarity analysis," *Nature*, Vol. 202, pp. 1034–1035, 1964.

[MacN 65] MacNaughton-Smith P. "Some statistical and other numerical techniques for classifying individuals," *Home Office Research Unit Report No.* 6, London: H.M.S.O., 1965.

[Marr 71] Marriot F.H.C. "Practical problems in a method of cluster analysis," *Biometrics*, Vol. 27, pp. 501–514, 1971.

[McQu 62] McQuitty L.L. "Multiple hierarchical classification of institutions and persons with reference to union-management relations and psychological well-being," *Educ. Psychol. Measur.*, Vol. 23, pp. 55–61, 1962.

[Mill 80] Milligan G.W. "An examination of the effect of six types of error perturbation on fifteen clustering algorithms," *Psychometrika*, Vol. 45, pp. 325–342, 1980.

[Mill 85] Milligan G.W., Cooper M.C. "An examination of procedures for determining the number of clusters in a data set," *Psychometrika*, Vol. 50(2), pp. 159–179, 1985.

[Mill 83] Milligan G.W., Soon S.C., Sokol L.M. "The effect of cluster size, dimensionality and the number of clusters on recovery of true cluster structure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 5(1), January 1983.

[Murt 83] Murtagh F. "A survey of recent advances in hierarchical clustering algorithms," *Journal of Computation*, Vol. 26, pp. 354–359, 1983.

[Murt 84] Murtagh F. "Complexities of hierarchic clustering algorithms: State of the art," *Computational Statistics Quarterly*, Vol. 1(2), pp. 101–113, 1984.

[Murt 85] Murtagh F. *Multidimensional clustering algorithms*, Physica-Verlag, COMPSTAT Lectures, Vol. 4, Vienna, 1985.

[Murt 95] Murthy M.N., Jain A.K. "Knowledge-based clustering scheme for collection, management and retrieval of library books," *Pattern Recognition*, Vol. 28(7), pp. 949–963, 1995.

[Olso 93] Olson C.F. "Parallel algorithms for hierarchical clustering," Technical Report, University of California at Berkeley, December. 1993.

[Ozaw 83] Ozawa K. "Classic: A hierarchical clustering algorithm based on asymmetric similarities," *Pattern Recognition*, Vol. 16(2), pp. 201–211, 1983.

[Prit 71] Pritchard N.M., Anderson A.J.B. "Observations on the use of cluster analysis in botany with an ecological example," *J. Ecol.*, Vol. 59, pp. 727–747, 1971.

[Rolp 73] Rolph F.J. "Algorithm 76: Hierarchical clustering using the minimum spanning tree," *Journal of Computation*, Vol. 16, pp. 93–95, 1973.

[Rolp 78] Rolph F.J. "A probabilistic minimum spanning tree algorithm," *Information Processing Letters*, Vol. 7, pp. 44–48, 1978.

[Same 89] Samet H. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, Boston, 1989.

[Shea 65] Sheals J.G. "An application of computer techniques to Acrine taxonomy: A preliminary examination with species of the Hypoaspio-Androlaelaps complex Acarina," *Proc. Linn. Soc. Lond.*, Vol. 176, pp. 11–21, 1965.

[Snea 73] Sneath P.H.A., Sokal R.R. *Numerical Taxonomy*, W.H. Freeman & Co., 1973.

[Solo 71] Solomon H. "Numerical taxonomy," in *Mathematics in the Archaeological and Historical Sciences* (Hobson F.R., Kendall D.G., Tautu P.A., eds.), University Press, 1971.

[Stri 67] Stringer P. "Cluster analysis of non-verbal judgement of facial expressions," *Br. J. Math. Statist. Psychol.*, Vol. 20, pp. 71–79, 1967.

[Will 89] Willett P. "Efficiency of hierarchic agglomerative clustering using the ICL distributed array processor," *Journal of Documentation*, Vol. 45, pp. 1–45, 1989.

[Will 77] Williams W.T., Lance G.N. "Hierarchical classificatory methods," in *Statistical Methods for Digital Computers* (Enslein K., Ralston A., Wilf H.S., eds.), John Wiley & Sons, 1977.

[Zhan 96] Zhang T., Ramakrishnan R., Livny M. "BIRCH: An efficient data clustering method for very large databases," *Proceedings of the ACM SIGMOD Conference on Management of Data*, pp. 103–114, Montreal, Canada, 1996.