Chapter 13. Cluster Analysis

Applied Multivariate Statistical Analysis

Course Instructor: Shuen-Lin Jeng

Department of Statistics
National Cheng Kung University

March, 28, 2021

13.1 The Problem

Cluster analysis is a set of tools for building groups (clusters) from multivariate data objects. The aim is to construct groups with homogeneous properties out of heterogeneous large samples. Cluster analysis can be divided into two fundamental steps.

1. *Choice of a proximity measure:*
   One checks each pair of observations (objects) for the similarity of their values. A similarity (proximity) measure is defined to measure the "closeness" of the objects. The "closer" they are, the more homogeneous they are.

2. *Choice of group-building algorithm:*
   On the basis of the proximity measures, the objects assigned to groups so that differences between groups become large and observations in a group become as close as possible.

13.2 The Proximity Between Objects

The starting point of a cluster analysis is a data matrix $\mathcal{X}(n \times p)$ with n measurements (objects) of p variables. The proximity (similarity) among objects is described by a matrix $\mathcal{D}(n \times n)$

$$\mathcal{D} = \begin{pmatrix} d_{11} & d_{12} & \cdots & \cdots & \cdots & d_{1n} \\ \vdots & d_{22} & & & & \vdots \\ \vdots & \vdots & \ddots & & & \vdots \\ \vdots & \vdots & & \ddots & & \vdots \\ \vdots & \vdots & & & \ddots & \vdots \\ d_{n1} & a_{n2} & \cdots & \cdots & \cdots & d_{nn} \end{pmatrix} \tag{13.1}$$

13.2 The Proximity Between Objects

- The matrix $\mathcal{D}$ contains measures of similarity or dissimilarity among the n objects. The values d_{ij} are distances, then they measure dissimilarity. If the values d_{ij} are proximity measures, the greater the proximity value, the more similar the objects are.
- A distance matrix, for example, could be defined by the L_2-norm: $d_{ij} = \left\| x_i - x_j \right\|_2$, where x_i and x_j denote the rows of the data matrix $\mathcal{X}$.

13.2 The Proximity Between Objects

If d_{ij} is a distance, then $d'_{ij} = \max_{i,j}\{d_{ij}\} - d_{ij}$ is a proximity measure. Nominal values (like binary variables) lead in general to proximity values, whereas metric values lead (in general) to distance matrices.

Similarity of Objects with Binary Structure

In order to measure the similarity between objects we always compare pairs of observations (x_i, x_j) where $x_i^\top = (x_{i1}, \dots, x_{ip})$, $x_j^\top = (x_{j1}, \dots, x_{jp})$, and x_{ik}, $x_{jk} \in \{0, 1\}$. Obviously there are four cases:

$$x_{ik} = x_{jk} = 1,$$

$$x_{ik} = 0, x_{jk} = 1,$$

$$x_{ik} = 1, x_{jk} = 0,$$

$$x_{ik} = x_{jk} = 0.$$

13.2 The Proximity Between Objects

Define

$$a_1 = \sum_{k=1}^{p} I\left(x_{ik} = x_{jk} = 1\right),$$

$$a_2 = \sum_{k=1}^{p} I\left(x_{ik} = 0, x_{jk} = 1\right),$$

$$a_3 = \sum_{k=1}^{p} I\left(x_{ik} = 1, x_{jk} = 0\right),$$

$$a_4 = \sum_{k=1}^{p} I\left(x_{ik} = x_{jk} = 0\right).$$

13.2 The Proximity Between Objects

Note that each a_l, $l = 1, \ldots, 4$, depends on the pair (x_i, x_j).

The following proximity measures are used in practice:

$$d_{ij} = \frac{a_1 + \delta a_4}{a_1 + \delta a_4 + \lambda (a_2 + a_3)} \tag{13.2}$$

where δ and λ are weighting factors.

13.2 The Proximity Between Objects

Table 13.1 The common similarity coefficients

Name	δ	λ	Definition
Jaccard	0	1	$\frac{a_1}{a_1+a_2+a_3}$
Tanimoto	1	2	$\frac{a_1+a_4}{a_1+2(a_2+a_3)+a_4}$
Simple Matching (M)	1	1	$\frac{a_1+a_4}{p}$
Russel and Rao (RR)	–	–	$\frac{a_1}{p}$
Dice	0	0.5	$\frac{2a_1}{2a_1+(a_2+a_3)}$
Kulczynski	–	–	$\frac{a_1}{a_2+a_3}$

13.2 The Proximity Between Objects

Distance Measures for Continuous Variables

A wide variety of distance measures can be generated by the L_r-norms, $r \geq 1$,

$$d_{ij} = \|x_i - x_j\|_r = \left\{ \sum_{k=1}^{p} |x_{ik} - x_{jk}|^r \right\}^{1/r} . \tag{13.3}$$

Here x_{ik} denotes the value of the k-th variable on object i.

The L_1-metric, for example, gives less weight to outliers than the L_2-norm (Euclidean norm).

13.2 The Proximity Between Objects

Example 13.2 Suppose we have $x_1 = (0,0)$, $x_2 = (1,0)$ and $x_3 = (5,5)$. Then the distance matrix for the L_1-norm is

$$\mathcal{D}_1 = \begin{pmatrix} 0 & 1 & 10 \\ 1 & 0 & 9 \\ 10 & 9 & 0 \end{pmatrix},$$

and for the squared L_2- or Euclidean norm

$$\mathcal{D}_2 = \begin{pmatrix} 0 & 1 & 50 \\ 1 & 0 & 41 \\ 50 & 41 & 0 \end{pmatrix}$$

One can see that the third observation x_3 receives much more weight in the squared L_2-norm than in L_1-norm.

13.2 The Proximity Between Objects

An underlying assumption in applying distance based on L_r-norms is that the variables are measured on the same scale. If this is not the case , a standardization should first be applied. This corresponds to using a more general L_2- or Euclidean norm with a metric $\mathcal{A}$, where $\mathcal{A} > 0$ (see Sect. 2.6):

$$d_{ij}^2 = \left\| x_i - x_j \right\|_{\mathcal{A}} = \left(x_i - x_j \right)^{\mathsf{T}} \mathcal{A} \left(x_i - x_j \right). \tag{13.4}$$

13.2 The Proximity Between Objects

L_2-norms are given by $\mathcal{A} = \mathcal{I}_p$, but if a standardization is desired, then the weight matrix $\mathcal{A} = \text{diag}\left(s_{X_1X_1}^{-1}, \ldots, s_{X_pX_p}^{-1}\right)$ may be suitable. Recall that $s_{X_kX_k}$ is the variance of the k-th component. Hence we have

$$d_{ij}^2 = \sum_{k=1}^{p} \frac{\left(x_{ik} - x_{jk}\right)^2}{s_{X_kX_k}}. \tag{13.5}$$

13.2 The Proximity Between Objects

If $\mathcal{X}$ is a contingency table, row i is characterized by the conditional frequency distribution $\frac{x_{ij}}{x_{i\bullet}}$, where $x_{i\bullet} = \sum_{j=1}^{p} x_{ij}$ indicates the marginal distributions over the rows: $\frac{x_{i\bullet}}{x_{\bullet\bullet}}$, $x_{\bullet\bullet} = \sum_{i=1}^{n} x_{i\bullet}$.

Similarly, column j of $\mathcal{X}$ is characterized by the conditional frequencies $\frac{x_{ij}}{x_{\bullet j}}$, where $x_{\bullet j} = \sum_{i=1}^{n} x_{ij}$. The marginal frequencies of the columns are $\frac{x_{\bullet j}}{x_{\bullet\bullet}}$.

13.2 The Proximity Between Objects

The distance between two rows, i_1 and i_2, corresponds to the distance between their respective frequency distributions. It is common to define this distance using the χ^2-metric:

$$d^2(i_1, i_2) = \sum_{j=1}^{p} \frac{1}{\left(\frac{x_{\bullet j}}{x_{\bullet\bullet}}\right)} \left(\frac{x_{i_1 j}}{x_{i_1 \bullet}} - \frac{x_{i_2 j}}{x_{i_2 \bullet}}\right)^2 \tag{13.7}$$

13.2 The Proximity Between Objects

Note that this can be expressed as a distance between the vectors $x_1 = \left(\frac{x_{i_1 j}}{x_{\bullet\bullet}}\right)$ and $x_2 = \left(\frac{x_{i_2 j}}{x_{\bullet\bullet}}\right)$ as in (13.4) with weighting matrix $\mathcal{A} = \left\{\text{diag}\left(\frac{x_{\bullet j}}{x_{\bullet\bullet}}\right)\right\}^{-1}$.

Similarly, if we are interested in clusters among the columns, we can define:

$$d^2\left(j_1, j_2\right) = \sum_{i=1}^{n} \frac{1}{\left(\frac{x_{i\bullet}}{x_{\bullet\bullet}}\right)} \left(\frac{x_{i j_1}}{x_{\bullet j_1}} - \frac{x_{i j_2}}{x_{\bullet j_2}}\right)^2 .$$

13.3 Cluster Algorithms

- There are essentially three traditional clustering methods: hierarchical and partitioning algorithms. The hierarchical algorithms can be divided into agglomerative and splitting procedures.
- The main difference between the two clustering techniques is that in hierarchical clustering once groups are found and elements are assigned to the groups, this assignment cannot be changed. In partitioning techniques, on the other hand, the assignment of objects into groups may change during the algorithm application.

13.3 Cluster Algorithms

Partitioning Algorithms

- Partitional clustering indicates a popular class of methods to find clusters in a set of data points. Here, the number of clusters k is fixed a priori. The points are embedded in a metric space, so that each vertex is a point and a distance measure is defined between pairs of points in the space.
- The most popular partitional technique in the literature is *k-means* clustering. The *k-means* standard algorithm is iterative and is starting from random partitions/points.

13.3 Cluster Algorithms

The objective function for this algorithm is the total intra-cluster distance or squared error function. The equation to be minimized is given in (13.10).

$$\hat{\mathcal{S}} = \arg\min_{\mathcal{S}} \sum_{j=1}^{k} \sum_{i \in \mathcal{S}_j} \left\| x_i - \mu_j \right\|^2 \tag{13.10}$$

with respect to $\mathcal{S} = \{S_1, \ldots, S_k\}$, $\bigcup_{j=1}^{k} S_j = \{1, 2, \ldots, n\}$, where S_j indicates the subset of points in cluster j and μ_j its centroid, k is the number of clusters, n is the number of data points.

13.3 Cluster Algorithms

- The *k-means* problem can be simply solved with the standard Lloyd's Algorithm 13.1. The solution found can be not optimal, and it strongly depends on the initial choice of the centroids. The result can be improved by performing more runs starting from different initial conditions, and picking the solution which yields the minimum value of the total intra-cluster distance.
- Other methods claiming to choose better initial centroids are *k-means* ++, *intelligent k-means* and *genetic k-means* (Arthur and Vassilvitskii 2007).

13.3 Cluster Algorithms

Algorithm 13.1 *k-means* Clustering Standard Algorithm

Fix an initial set $\{\mu_j^{(t)}\}_{j=1}^k$, $t = 1$

Assign $\hat{j}(i) = \underset{j}{\operatorname{argmin}} \|x_i - \mu_j^{(t)}\|^2$

x_i belongs then to cluster $\hat{j}(i)$ resulting in (new) partition

$$\bigcup_{j=1}^{k} \mathcal{S}_j^{(t)} = \{1, \ldots, n\}$$

Update $\mu_j^{(t+1)} = \left(\#\mathcal{S}_j^{(t)}\right)^{-1} \sum_{i \in \mathcal{S}_j^{(t)}} x_i$

repeat

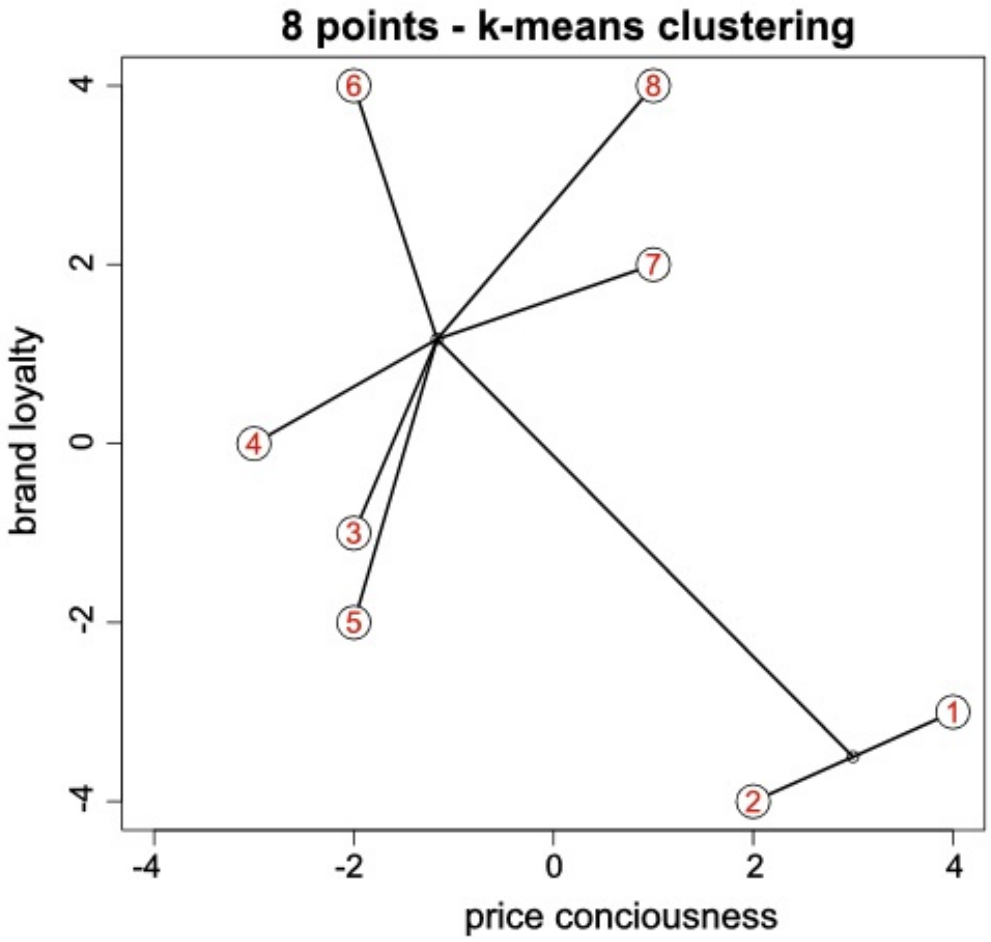
assign, update

until convergence in terms of (13.10).

13.3 Cluster Algorithms

Fig. 13.1 8 points—*k-means*—clustering

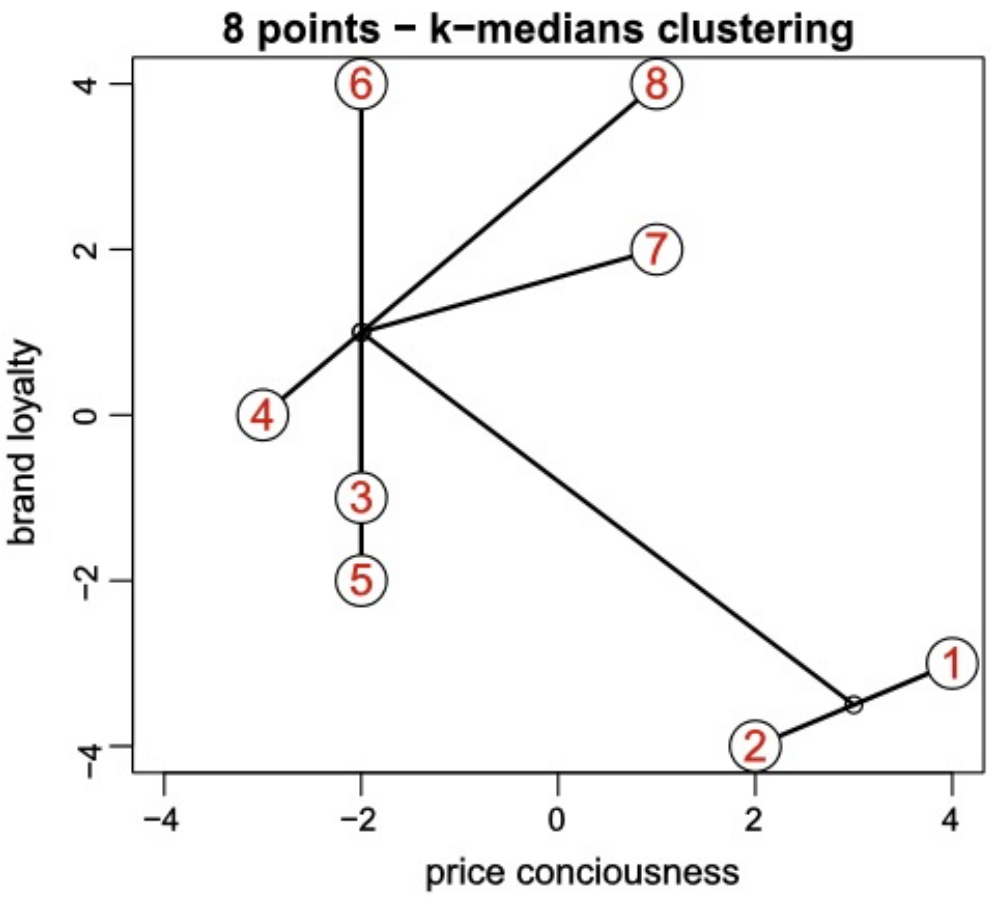
MVAclus8km



13.3 Cluster Algorithms

Fig. 13.2 8 points—*k-median*—clustering

MVAclus8km



13.3 Cluster Algorithms

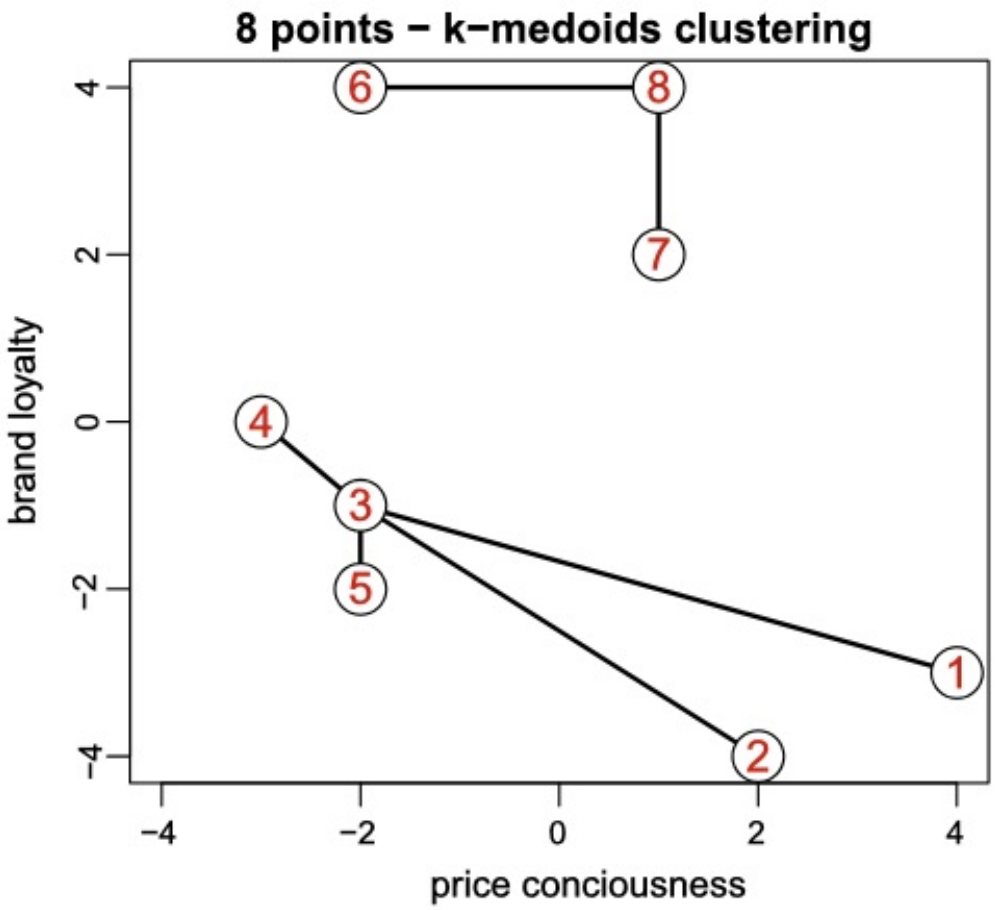
Some of other popular partitioning methods are listed below:

- *k-medoids*: for each cluster j one defines a reference point with position x_j, the most centrally located object in the cluster — medoid. Therefore, the difference between *k-means* and *k-medoids* is *k-means* can select the k virtual centroids, for *k-medoids* centroids should be k representatives of real objects.

13.3 Cluster Algorithms

Fig. 13.3 8 points—*k-medoids*—clustering

MVAclus8km



13.3 Cluster Algorithms

- *k-mode*: is an extension to handle with categorical data sets, by replacing means of clusters with modes.
- *k-median*: algorithm was developed to overcome the issue of *k-means'* high sensitivity to outliers, because empirical mean is easily influenced by extremes. Instead of calculating the mean for each cluster to determine its centroid, one calculates the median with respect to coordinates(Fig. 13.2). The objective function uses Manhattan distance instead of squared Euclidean distance and therefore (13.10) can be reformulated for this algorithm as follows:

$$\hat{\mathcal{S}} = \arg\min_{\mathcal{S}} \sum_{j=1}^{k} \sum_{i \in \mathcal{S}_j} \left| x_i - med_j \right| \tag{13.11}$$

13.3 Cluster Algorithms

- Another popular technique, similar to *k-means* clustering, is *fuzzy k -means*. This method that allows each data point to belong to multiple clusters with varying degrees of membership (Bezdek 1981).
- It should be noted though that there are some drawbacks of partition clustering methods. The number of clusters k must be preset at the beginning, which might lead to a biased clustering in the end. Also it is important to realize that the presented technology cannot deal with clusters of non-convex shape.

13.3 Cluster Algorithms

Recently developed method *Adaptive Weights Clustering(AWC)*, which is close in spirit *fuzzy* clustering, allows to overcome mentioned limitations of partitioning methods.

Hierarchical Algorithms, Agglomerative Techniques

Algorithm 13.3 Hierarchical Algorithms—Agglomerative Technique

```
Construct the finest partition
Compute the distance matrix 𝒟.
repeat
    Find the two clusters with the closest distance
    Put those two clusters into one cluster
    Compute the distance between the new groups and obtain a reduced distance matrix 𝒟
until all clusters are agglomerated into 𝒳
```

13.3 Cluster Algorithms

- If two objects or groups say, P and Q, are united, one computes the distance between this new group (object) $P+Q$ and group R using the following distance function:

$$d(R,P+Q) = \delta_1 d(R,P) + \delta_2 d(R,Q) + \delta_3 d(P,Q) + \delta_4 \left| d(R,P) - d(R,Q) \right| \tag{13.13}$$

- The δ_j's are weighting factors that lead to different agglomerative algorithms as described in Table 13.2. Here $n_P = \sum_{i=1}^{n} I\left(x_i \in P\right)$ is the number of objects in group P. The values of n_Q and n_R are defined analogously.

13.3 Cluster Algorithms

Table 13.2 Computations of group distances

Name	δ_1	δ_2	δ_3	δ_4
Single linkage	1/2	1/2	0	−1/2
Complete linkage	1/2	1/2	0	1/2
Average linkage (unweighted)	1/2	1/2	0	0
Average linkage (weighted)	$\frac{n_P}{n_P+n_Q}$	$\frac{n_Q}{n_P+n_Q}$	0	0
Centroid	$\frac{n_P}{n_P+n_Q}$	$\frac{n_Q}{n_P+n_Q}$	$-\frac{n_P n_Q}{(n_P+n_Q)^2}$	0
Median	1/2	1/2	−1/4	0
Ward	$\frac{n_R+n_P}{n_R+n_P+n_Q}$	$\frac{n_R+n_Q}{n_R+n_P+n_Q}$	$-\frac{n_R}{n_R+n_P+n_Q}$	0

13.3 Cluster Algorithms

Algorithm 13.4 Modified Hierarchical Algorithms—Agglomerative Technique

1: Construct the finest partition
2: Compute the distance matrix $\mathcal{D}$.
3: **repeat**
4: Find the smallest (Single linkage)/ largest (Complete linkage) value d (between objects m and n) in $\mathcal{D}$
5: If m and n are not in the same cluster, combine the clusters m and n belonging to together, and delete the smallest value
6: **until** all clusters are agglomerated into $\mathcal{X}$ or the value d exceeds the preset level

As instead of computing new distance matrices every step, a linear search in the original distance matrix is enough for clustering in the modified algorithm, it is more efficient in practice.

13.3 Cluster Algorithms

Example 13.6 Let us examine the agglomerative algorithm for three points in Example 13.2, $x_1 = (0, 0)$, $x_2 = (1, 0)$ and $x_3 = (5, 5)$, and the squared Euclidean distance matrix with single linkage weighting. The algorithm starts with $N = 3$ clusters: $P = \{x_1\}$, $Q = \{x_2\}$ and $R = \{x_3\}$. The distance matrix $\mathcal{D}_2$ is given in Example 13.2. The smallest distance in $\mathcal{D}_2$ is the one between the clusters P and Q. Therefore, applying step 4 in the above algorithm we combine these clusters to form $P + Q = \{x_1, x_2\}$.

13.3 Cluster Algorithms

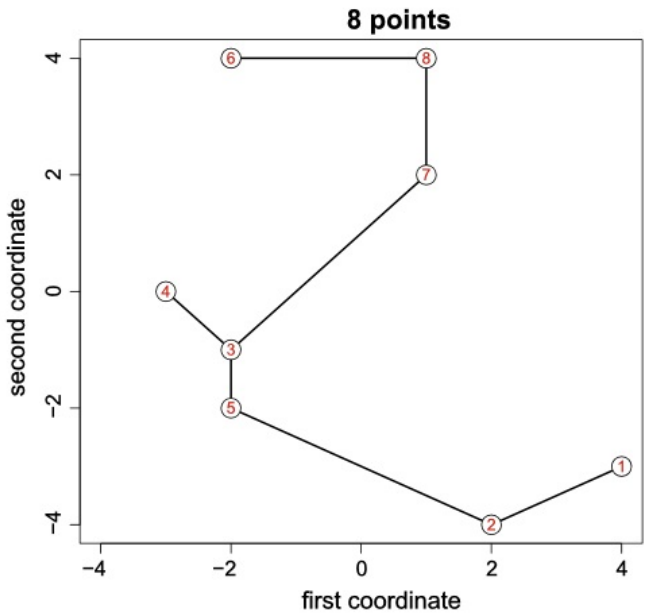
The single linkage distance between the remaining two clusters is from Table 13.2 and (13.13) equal to

$$d(R, P+Q) = \frac{1}{2}d(R,P) - \frac{1}{2}d(R,Q) - \frac{1}{2}\left|d(R,P) + d(R,Q)\right| \quad (13.14)$$

$$\begin{aligned} &= \tfrac{1}{2}d_{13} + \tfrac{1}{2}d_{23} - \tfrac{1}{2}\cdot|d_{13} - d_{23}| \\ &= \tfrac{50}{2} + \tfrac{41}{2} - \tfrac{1}{2}\cdot|50 - 41| \\ &= 41. \end{aligned}$$

13.3 Cluster Algorithms

Fig. 13.4 The 8-point example MVAclus8p

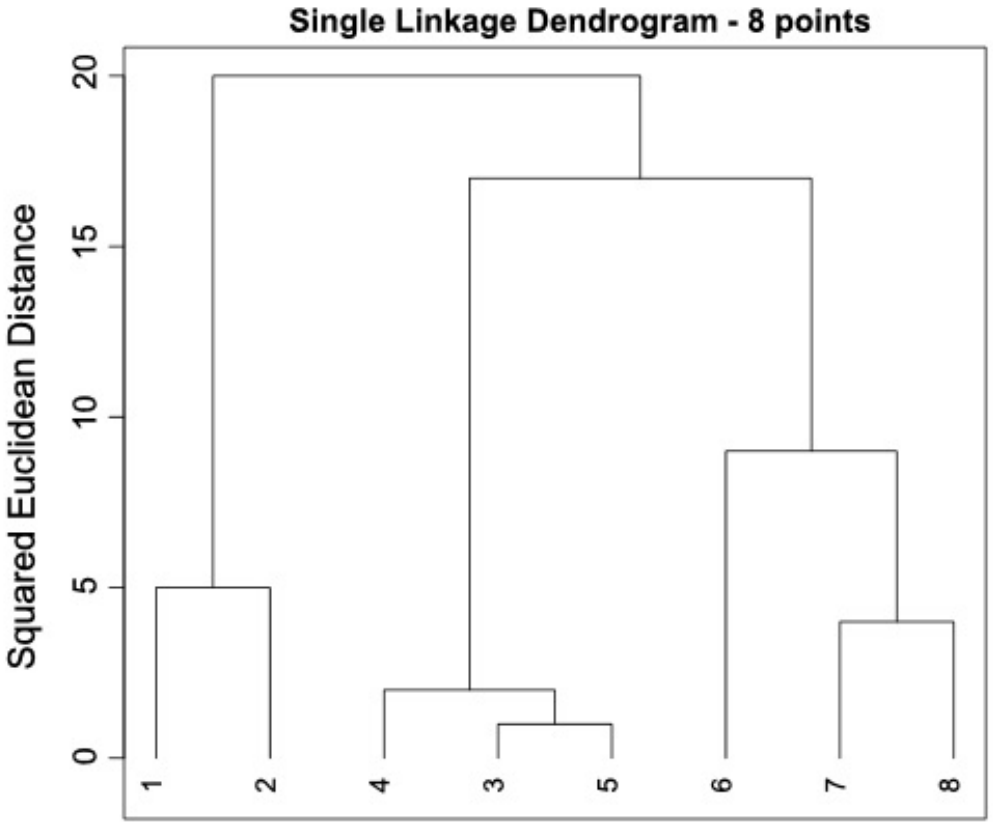


The reduced distance matrix is then $\begin{pmatrix} 0 & 41 \\ 41 & 0 \end{pmatrix}$. The next and last step is to unite the clusters R and $P + Q$ into a single cluster $\mathcal{X}$, the original data matrix. A graphical representation of the sequence of clustering is called a *dendrogram*.

13.3 Cluster Algorithms

Fig. 13.5 The dendrogram for the 8-point example, Single linkage algorithm

MVAclus8p



13.3 Cluster Algorithms

- The *single linkage* algorithm defines the distance between two groups as the smallest value of the individual distances. Table 13.2 shows that in this case

$$d(R, P + Q) = \min\{d(R, P), d(R, Q)\} \tag{13.15}$$

- This algorithm is also called the *Nearest Neighbor* algorithm. As a consequence of its construction, single linkage tends to build large groups.

13.3 Cluster Algorithms

- The *complete linkage* algorithm tries to correct this kind of grouping by considering the largest (individual) distances. Indeed, the complete linkage distance can be written as

$$d(R, P + Q) = \max\{d(R, P), d(R, Q)\}. \tag{13.16}$$

- It is also called the *Farthest Neighbor* algorithm. This algorithm will cluster groups where all the points are proximate, since it compares the largest distances.

13.3 Cluster Algorithms

The *average linkage* algorithm (weighted or unweighted) proposes a compromise between the two preceding algorithms, in that it computes an average distance:

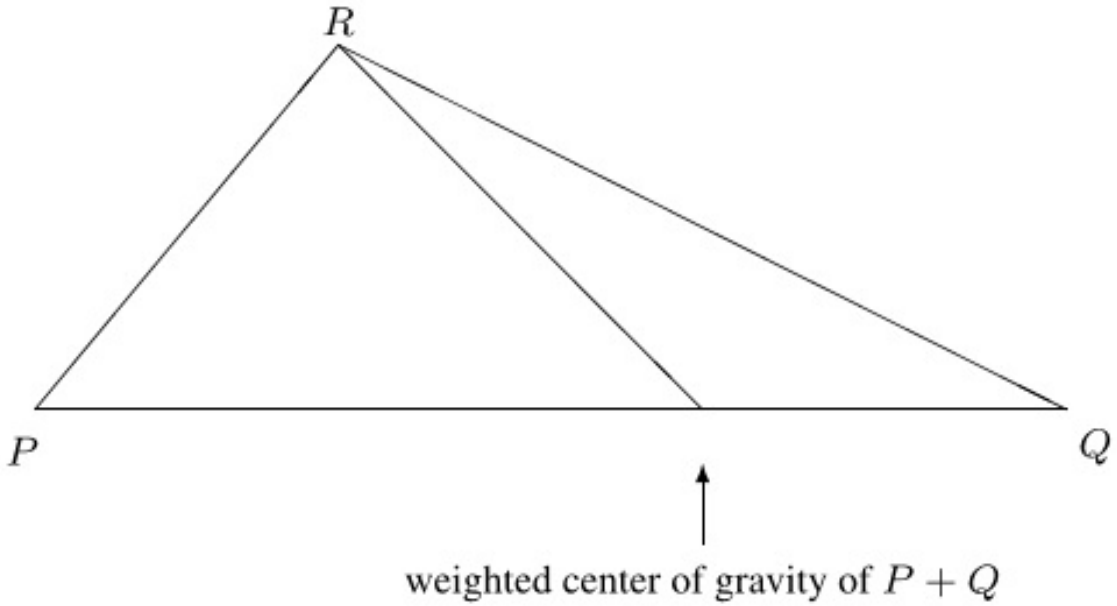
$$d(R, P+Q) = \frac{n_P}{n_P + n_Q} d(R, P) + \frac{n_Q}{n_P + n_Q} d(R, Q) \tag{13.17}$$

The *centroid* algorithm is quite similar to the average linkage algorithm and uses the natural geometrical distance between R and the weighted center of gravity of P and Q (see Fig. 13.6):

$$d(R, P+Q) = \frac{n_P}{n_P + n_Q} d(R, P) + \frac{n_Q}{n_P + n_Q} d(R, Q) - \frac{n_P n_Q}{\left(n_P + n_Q\right)^2} d(P, Q) \tag{13.18}$$

13.3 Cluster Algorithms

Fig. 13.6 The centroid algorithm



13.3 Cluster Algorithms

- The *Ward clustering* algorithm computes the distance between groups according to the formula in Table 13.2.
- The main difference between this algorithm and the linkage procedures is in the unification procedure. The Ward algorithm does not put together groups with smallest distance. Instead, it joins groups that do not increase a given measure of heterogeneity "too much".
- The aim of the Ward procedure is to unify groups such that the variation inside these groups does not increase too drastically: the resulting groups are as homogeneous as possible.

13.3 Cluster Algorithms

- The heterogeneity of group R is measured by the inertia inside the group. This inertia is defined as follows:

$$I_R = \frac{1}{n_R} \sum_{i=1}^{n_R} d^2 \left(x_i, \bar{x}_R \right) \tag{13.19}$$

  where $\bar{x}_R$ is the center of gravity (mean) over the groups. I_R clearly provides a scalar measure of the dispersion of the group around its center of gravity.

- If the usual Euclidean distance is used, then I_R represents the sum of the variances of the p components of x_i inside group R.

13.3 Cluster Algorithms

- When two objects or groups P and Q are joined, the new group $P + Q$ has a larger inertia I_{P+Q}. It can be shown that the corresponding increase of inertia is given by

$$\Delta(P, Q) = \frac{n_P n_Q}{n_P + n_Q} d^2(P, Q) \tag{13.20}$$

- The Ward algorithm is defined as an algorithm that "joins the groups that give the smallest increase in $\Delta(P, Q)$".
- The Ward algorithm is related to the centroid algorithm, but with an "inertial" distance Δ rather than the "geometric" distance d^2.

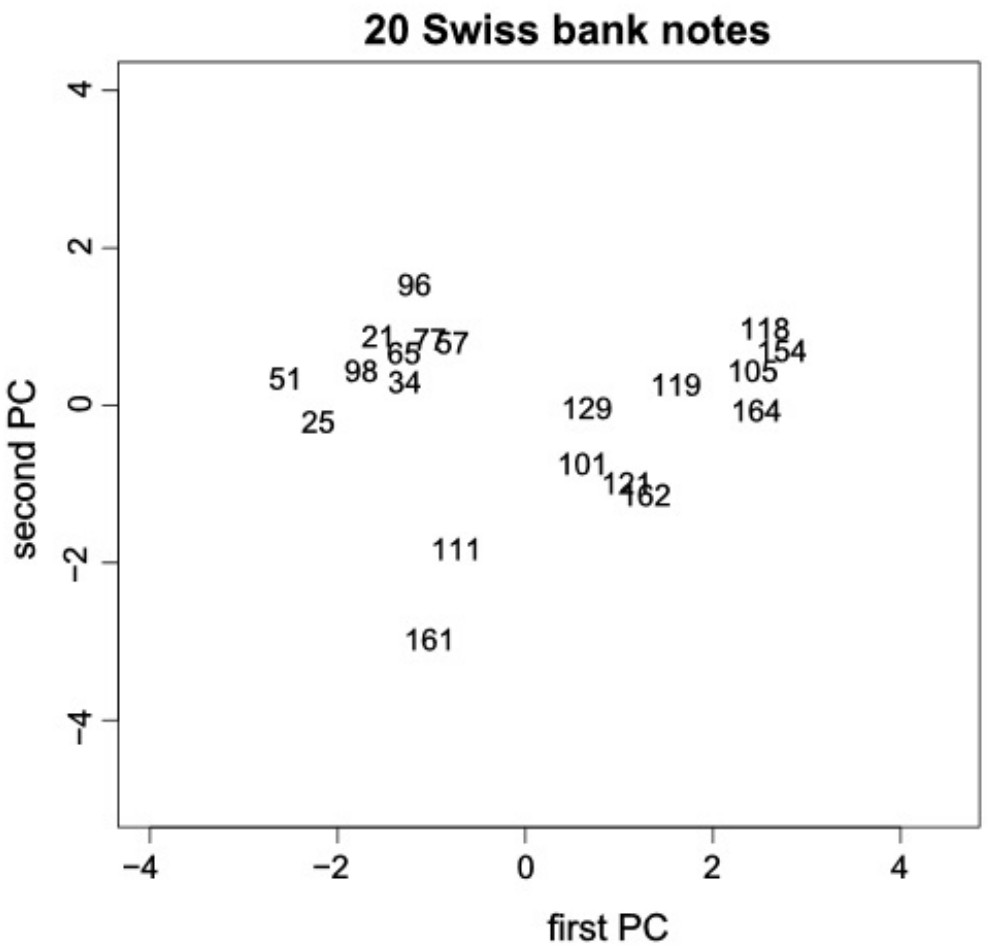
13.3 Cluster Algorithms

Example 13.8 As an example we randomly select 20 observations from the bank notes data and apply the Ward technique using Euclidean distances. Figure 13.7 shows the first two PCs of these data, Fig. 13.8 displays the dendrogram.

13.3 Cluster Algorithms

Fig. 13.7 PCA for 20 randomly chosen bank notes

MVAclusbank



13.3 Cluster Algorithms

Example 13.9 Consider the French food expenditures data set. As in Chap. 11 we use the standardized data which is equivalent to $\mathcal{A} = \text{diag}\left(s_{X_1X_1}^{-1}, \ldots, s_{X_7X_7}^{-1}\right)$ as the weight matrix in the L_2-norm. The NPCA plot of the individuals was given in Fig. 11.7. The Euclidean distance matrix is of course given by (13.6). The dendrogram obtained by using the Ward algorithm is shown in Fig. 13.9.

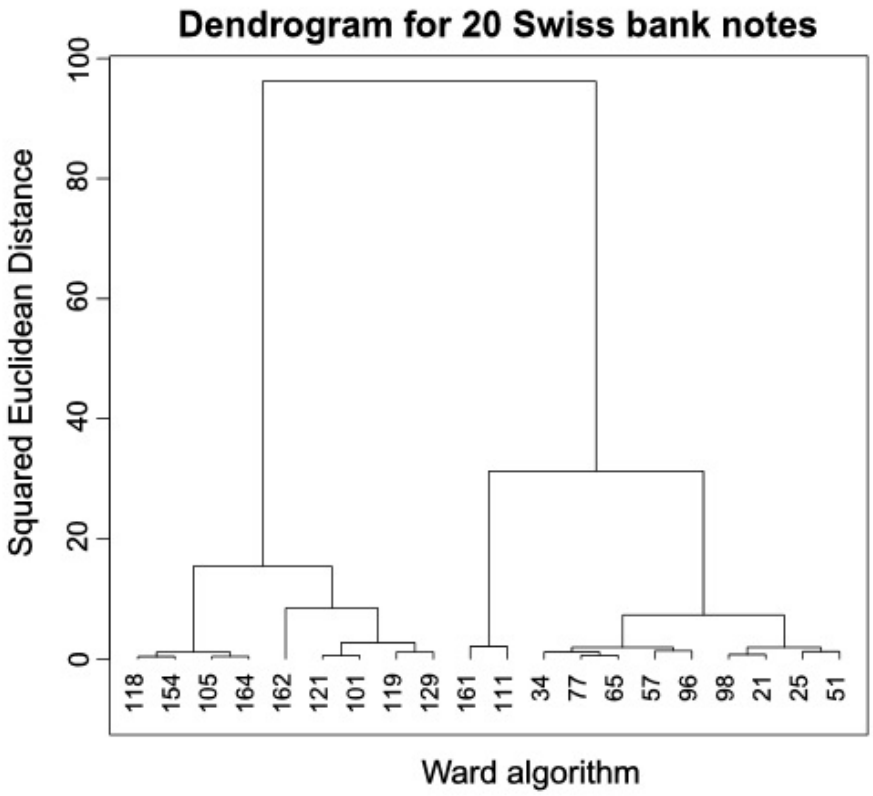
13.3 Cluster Algorithms

- If the aim was to have only two groups, as can see in Fig. 13.9, they would be {CA2, CA3, CA4, CA5, EM5} and {MA2, MA3, MA4, MA5, EM2, EM3, EM4}.
- Clustering three groups is somewhat arbitrary (the levels of the distance are too similar).
- If we were interested in four groups, we would obtain { CA2, CA3, CA4}, { EM2, MA2, EM3, MA3}, {EM4, MA4, MA5}, and {EM5, CA5}.

  This grouping shows a balance between socio-professional levels and size of the families in determining the clusters. The four groups are clearly well represented in the NPCA plot in Fig. 11.7.

13.3 Cluster Algorithms

Fig. 13.8 The dendrogram for the 20 bank notes, Ward algorithm MVAclusbank



13.4 Adaptive Weights Clustering

- An alternative clustering technique may be based on nonparametric ideas of finding cluster structure through a separation approach via a homogeneity detection test. We follow here the exposition of Efimov et al. (2017).
- The method is fully adaptive and does not require to specify the number of clusters or their structure.
- The clustering results are not sensitive to noise and outliers, the procedure is able to recover different clusters with sharp edges or manifold structure.

13.4 Adaptive Weights Clustering

- Let $\{X_i\}_{i=1}^n \subset \mathbb{R}^p$. The proposed procedure operates with the distance (or similarity) matrix $\mathcal{D} = \left(d\left(X_i, X_j\right)\right)_{i,\ \mathrm{j}=1}^n$ only. For describing the clustering structure of the data, we introduce a $n \times n$ matrix of weights $W = \left(w_{ij}\right)$, $i, j = 1, \ldots, n$.
- Usually the weights w_{ij} are binary and $w_{ij} = 1$ means that X_i and X_j are in the same cluster, while $w_{ij} = 0$ indicates that these points are in different clusters.

13.4 Adaptive Weights Clustering

- The matrix W should be symmetric and each block of ones describes one cluster.
- Below we do not require a block structure which allows to incorporate even overlapping clusters.
- For every fixed i, the associated cluster C_i is given by the collection of positive weights $\left(w_{ij}\right)$ overall j.

13.4 Adaptive Weights Clustering

Sequence of radii:

One looks at a growing sequence of radii $h_1 \leq h_2 \leq ... \leq h_K$ which determines how fast the algorithm will evolve from considering local structures to large-scale objects.

Initialization of weights:

On initialization step we connect each point with its n_0 closest neighbors:

$$w_{ij}^{(0)} = I\left[d\left(X_i, X_j\right) \leq \max\left\{h_0\left(X_i\right), h_0\left(X_j\right)\right\}\right], \tag{13.21}$$

where $h_0\left(X_i\right)$ is the distance between X_i and its n_0 closest neighbor, our default choice $n_0 = 2p + 2$.

13.4 Adaptive Weights Clustering

Updates at step k

Fig. 13.10 Test of "no gap between local clusters". From left: Homogeneous case; $N_{i\wedge j}^{(k)}$; $N_{i\Delta j}^{(k)}$; $N_{i\vee j}^{(k)}$

The empirical mass of the overlap $N_{i\wedge j}^{(k)}$ can be naturally defined as

$$N_{i\wedge j}^{(k)} = \sum_{\ell \neq i,\, \mathrm{j}} w_{il}^{(k-1)} w_{jl}^{(k-1)}. \tag{13.22}$$

13.4 Adaptive Weights Clustering

- Similarly, the *mass of the complement* is defined as

$$N_{i\triangle j}^{(k)} = \sum_{I \neq i,j} \left\{ w_{il}^{(k-1)} I\left(X_l \notin B\left(X_j, h_{k-1}\right)\right) + w_{jl}^{(k-1)} I\left(X_l \notin B\left(X_j, h_{k-1}\right)\right) \right\} \tag{13.23}$$

  Note that $N_{i\triangle j}^{(k)}$ is nearly the number of points in C_i^{k-1} and C_j^{k-1} which do not belong to the overlap $B\left(X_i, h_{k-1}\right) \cap B\left(X_j, h_{k-1}\right)$.
- Finally, *mass of the union* $N_{i\vee j}^{(k)}$ can be defined as the sum of the mass of overlap and the mass of the complement:

$$N_{i\vee j}^{(k)} = N_{i\wedge j}^{(k)} + N_{i\triangle j}^{(k)}. \tag{13.24}$$

  To measure the gap, consider the ratio of these two masses:

$$\tilde{\theta}_{ij}^{(k)} = N_{i\wedge j}^{(k)} / N_{i\vee j}^{(k)} \tag{13.25}$$

13.4 Adaptive Weights Clustering

- To quantify the notion of significance, we consider the statistical likelihood ratio test of "no gap" between two local clusters, that is $\tilde{\theta}_{ij}^{(k)} > q_{ij}^{(k)}$ vs $\tilde{\theta}_{ij}^{(k)} \leq q_{ij}^{(k)}$:

$$T_{ij}^{(k)} = N_{i \vee j}^{(k)} KL\left(\tilde{\theta}_{ij}^{(k)}, q_{ij}^{(k)}\right)\left\{I\left(\tilde{\theta}_{ij}^{(k)} \leq q_{ij}^{(k)}\right) - I\left(\tilde{\theta}_{ij}^{(k)} > q_{ij}^{(k)}\right)\right\}. \tag{13.27}$$

- $KL(\theta, \eta)$ is the *Kullback-Leibler(KL) divergence* between two Bernoulli laws with parameters θ and η:

$$KL(\theta, \eta) = \theta \log \frac{\theta}{\eta} + (1-\theta) \log \frac{1-\theta}{1-\eta} \tag{13.28}$$

13.4 Adaptive Weights Clustering

- In the end, we update the weights $w_{ij}^{(k)}$ for all pairs of points X_i and X_j with distance $d\left(X_i, X_j\right) \leq h_k$:

$$w_{ij}^{(k)} = I\left(T_{ij}^{(k)} \leq \lambda\right). \tag{13.29}$$

- The first indicator allows us to recompute only $n \times n_k$ weights, where n_k is the average number of neighbors in the h_k neighborhood.
- Test $T_{ij}^{(k)}$ are scaled by a global constant λ which is the only tuning parameter of the method.

13.4 Adaptive Weights Clustering

Parameter tuning

A heuristic choice of λ is proposed based on the effective cluster size given by the total sum of final weights w_{ij}^{K}.

Algorithm 13.5 AWC

1: **Fix** a sequence of radii $h_1 \leq h_2 \leq \ldots \leq h_K$
2: **Initialization of weights**: $w_{ij}^{(0)} = \mathrm{I}\left(d(X_i, X_j) \leq \max(h_0(X_i), h_0(X_j))\right)$
3: **Updates at step** k :
4: Compute $T_{ij}^{(k)}$ using 13.27
5: $w_{ij}^{(k)} = \mathrm{I}\left(d(X_i, X_j) \leq h_k\right) \mathrm{I}\left(T_{ij}^{(k)} \leq \lambda\right)$
6: **Repeat** until $k = K$.

13.5 Spectral Clustering

- *Spectral clustering* is based on a graph theoretic approach to divide the data points into homogeneous groups (clusters).
- All techniques are based on the observations $\{x_i\}_{i=1}^n \in \mathbb{R}^p$ that are collected in the data matrix $\mathcal{X}(n \times p)$ and their similarity (13.2), between all data points.

13.5 Spectral Clustering

- The graph theoretic point of view on $\mathcal{D}$ is based on the undirected graph
  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the vertices $v_i \in \mathcal{V}$ correspond to the data points x_i.
- Two v's are connected if $d_i > 0$ and the edge between v_i and v_j is weighted by d_i.
- Clustering now boils down to find a partition of $\mathcal{G}$ such that the edges e_{ij} between different groups have low weights, in terms of values of the $\mathcal{D}$ matrix.
- The similarity matrix can be also called an adjacency matrix $\mathcal{W}$ of $\mathcal{G}$, since with element w_{ij} of $\mathcal{W}$ the fact that $w_{ij} = 0$ corresponds the notes v_i and v_j being not connected (or not similar) (Fig. 13.12).

13.5 Spectral Clustering

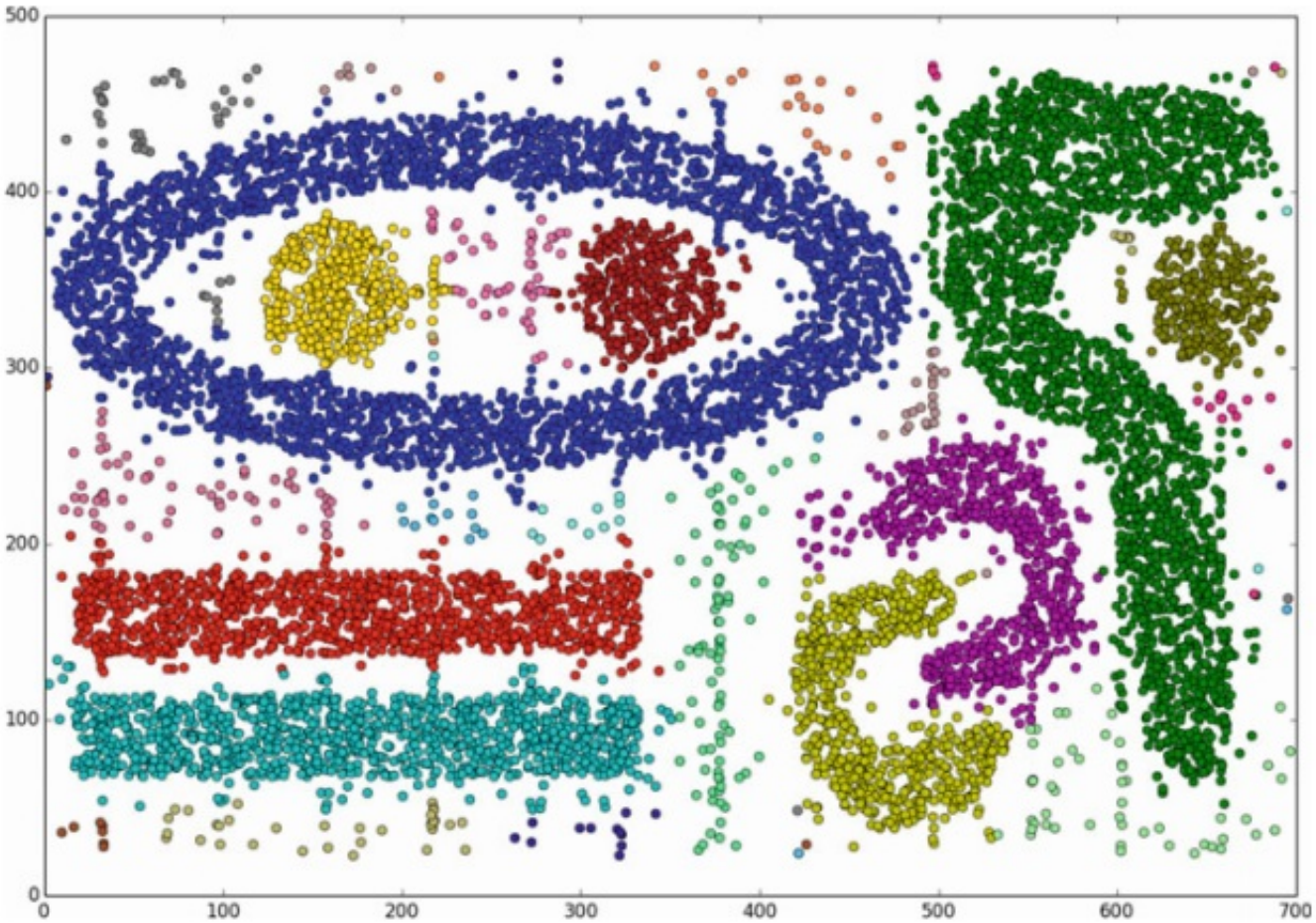


Fig. 13.12 AWC performance for $\lambda = 15$

13.5 Spectral Clustering

- Since clusters are different through local closeness of data points, it makes sense to study the connectedness through the degree $d_i = \sum_{j=1}^{n} w_{ij}$.
- Note that $d_{ii} = w_{ii}$ makes x_i a singleton, since it is not connected to any other vector.
- Define $\mathcal{D} = \text{diag}(d_i)$ and indicator vector $l_A = (f_1, \ldots, f_n)^T$, $f_i = I(x_i \in A)$, $A \subset \mathcal{V}$.
- The size of subset $A \subset \mathcal{V}$ is measured via $|A| \overset{def}{=}$ # of vertices on A $vol(A) \overset{def}{=} \sum_{i \in A} d_i$. If $\bar{A} = \mathcal{V} \backslash A$ then we can define A to be connected if there no points w_{ij} for $i \in A$, $j \in \bar{A}$.

13.5 Spectral Clustering

The basic idea of spectral clustering is a skillful eigenanalysis of $\mathcal{L} = \mathcal{D} - -\mathcal{W}$, the so-called Laplacian matrix.

Laplacian matrix properties:

1. $\forall f \in \mathbb{R}^n$, $f^\top \mathcal{L} = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} \left(f_i - f_j\right)^2$
2. 0 is the smallest eigenvalue of L with eigenvector l_n
3. $\mathcal{L}$ is symmetric and positive semi-definite
4. $\mathcal{L}$ has n eigenvalues $0 = \lambda_1 \leq ... \leq \lambda_n$

The unnormalized Laplacian gives us a tool to detect connected components.

13.5 Spectral Clustering

- Let $\mathcal{G}$ be an undirected graph with nonnegative weights. Then the multiplivitz k of the eigenvalue 0 of $\mathcal{L}$ equals the number of connected components $A_1, \ldots, A_k$ in the graph.
- The eigenspace of eigenvalue 0 is spanned by the indicator vectors $l_{A_1}, \ldots, l_{A_k}$ of those components. In fact one can describe k-component structure of $\mathcal{G}$ as a block diagonal structure of $\mathcal{L}$.

13.5 Spectral Clustering

- The normalized Laplacian is defined as

$$\mathcal{L}_1 \overset{def}{=} \mathcal{D}^{-\frac{1}{2}}\mathcal{L}\mathcal{D}^{-\frac{1}{2}} = \mathcal{I}_n - \mathcal{D}^{-\frac{1}{2}}\mathcal{W}\mathcal{D}^{-\frac{1}{2}} \tag{13.31}$$

- It is not hard to see that in modification of properties 1 and 2 we have:

$$f^T\mathcal{L}f = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}\left(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}}\right)^2 \tag{13.32}$$

- And the number of connected components can be identified via the multiplying k of the eigenvalues of $\mathcal{L}_1$ with eigenspace spanned by $\mathcal{D}^{\frac{1}{2}}1_{A_i}, i = 1, \ldots, k$.

13.5 Spectral Clustering

The normalized spectral clustering algorithm is built the same way as for an unnormalized case, except that at *step 3* one solves the eigenvalue problem for $\mathcal{L}_1$ (13.31) and on *step 4* gets changed to building the $\mathcal{U}(n \times p)$ object from $u_{ij} = v_{ij} / \left(\sum_k u_{ij}^2 \right)^{\frac{1}{2}}$.

13.5 Spectral Clustering

Algorithm 13.6 Spectral clustering algorithm

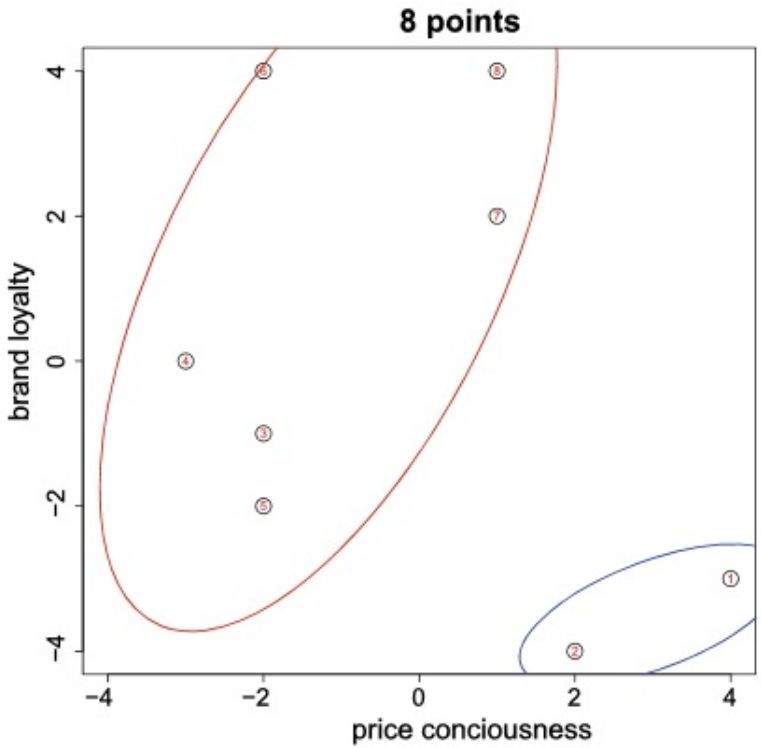
1: Construct from the proximity matrix the adjacency matrix $\mathcal{W}$
2: Compute unnormalized Laplacian $\mathcal{L}$.
3: Compute the first k eigenvectors $\nu_1, \ldots, \nu_k$ of $\mathcal{L}$
4: Define the $(n \times k)$ matrix $\mathcal{V} = (\nu_1, \nu_2 \ldots, \nu_k)$ with ν_i as columns
5: Take $y_i \in \mathbb{R}^k$ a the i-th row of $\mathcal{V}$, yielding the data matrix $\mathcal{Y}(n \times p)$
6: Cluster $\mathcal{Y}$ according to the *k-means* into $C_1, \ldots C_k$.
7: Output: Clusters $A_1, \ldots, A_k$ with $A_i = \{j : y_j \in C_j\}$

13.5 Spectral Clustering

Example 13.10 Figure 13.13 demonstrates partition to two clusters of 8 points performed by *spectral clustering algorithm.*

Fig. 13.13 The 8 points example; second smallest eigenvalue: −0.98 and corresponding eigenvector $(-0.0000, 0.0000, -0.0001, 0.0006, 0.0000, -0.1605, -0.6857, 0.7099)^\top$

MVAclus8psc



13.5 Spectral Clustering

The graph cut point of view on clustering can be most easily explained for $k = 2$.

Define

$$cut(A, B) = \sum_{i \in A,\ j \in B} w_{ij}$$

for two sets and

$$cut\left(A_1, \ldots, A_k\right) = \sum_{i=1}^{k} cut\left(A_i, \bar{A}_i\right)$$

13.5 Spectral Clustering

- In order to avoid singletons as group, one likes to create reasonably large group.
- This is achieved by measuring the weights of the edges via

$$Ncut = \sum_{i=1}^{k} \frac{cut\left(A_i, \bar{A}_i\right)}{vol\left(A_i\right)} \tag{13.33}$$

- A related measure RatioCut is based on scaling with $|A_i|$. How are these penalizations related to $\mathcal{L}_1$? Suppose that one likes to build (k = 2)

$$A \overset{\arg\min}{\subset} \mathcal{V} RatioCut(A, \bar{A}).$$

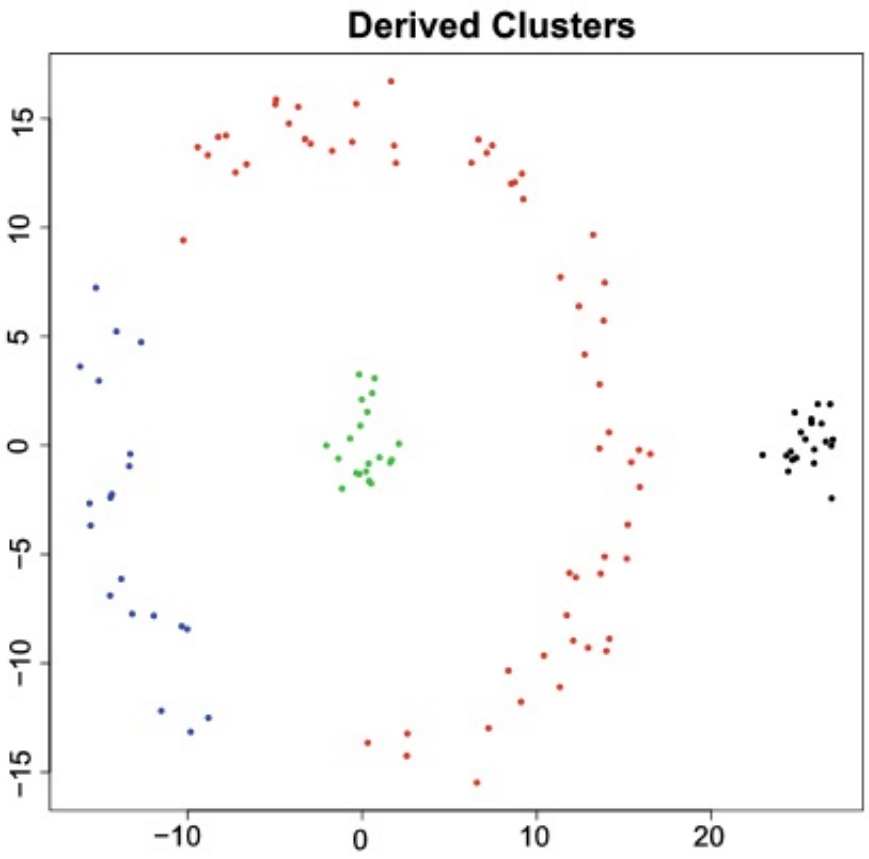
The Objective function can be written as

$$f^{\mathsf{T}} \mathcal{L} f = |\mathcal{V}| \cdot RatioCut(A, \bar{A}) \tag{13.34}$$

13.5 Spectral Clustering

Fig. 13.14 Parcellation results for the simulated data into four clusters by NCut algorithm based on the Euclidean distance

MVAspecclust



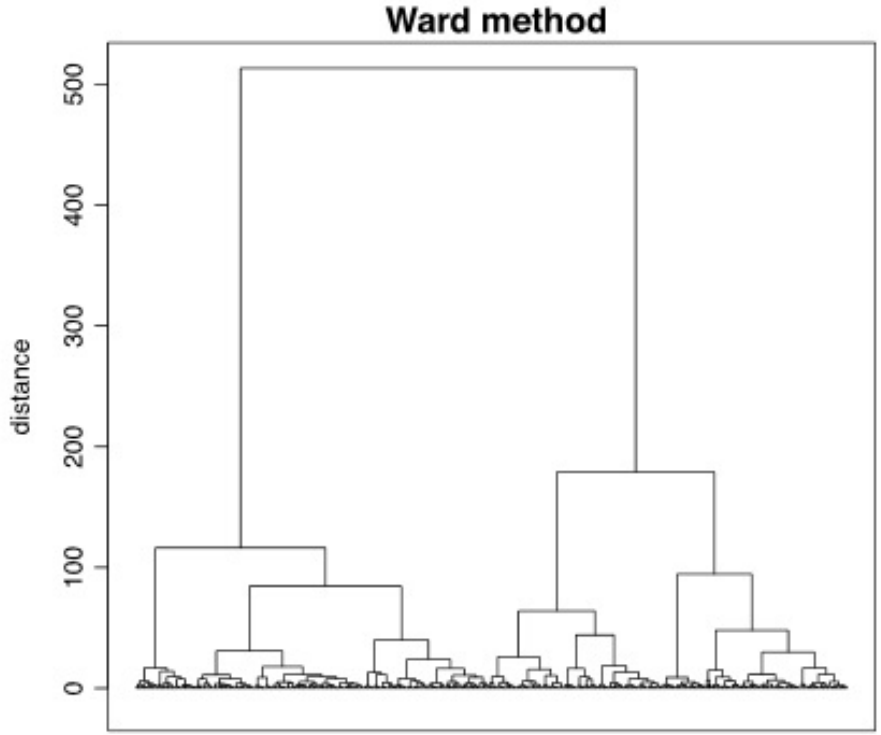
13.6 Boston Housing

- We focus on our attention to 14 transformed and standardized variables, see, e.g., Fig. 13.15 that provides descriptive statistics via boxplots for two clusters, as discussed in the sequel.
- A dendrogram for 13 variables (excluding the dummy variable $\widehat{X}_4$ — Charles River indicator) using the Ward method is displayed in Fig. 13.16.

13.6 Boston Housing

Fig. 13.16 Dendrogram of the Boston housing data using the Ward algorithm

MVAclusbh



13.6 Boston Housing

- To interpret the two clusters, we present the mean values and their respective standard errors of the 13 $\tilde{\mathcal{X}}$ variables by groups in Table13.3.
- Comparison of the mean values for both groups shows that all the differences in the means are individually significant.

13.6 Boston Housing

Table 13.3 Means and standard errors of the 13 standardized variables for Cluster 1 (251 observations) and Cluster 2 (255 observations). MVAclusbh

Variable	Mean C1	*SE* C1	Mean C2	*SE* C2
1	−0.7105	0.0332	0.6994	0.0535
2	0.4848	0.0786	−0.4772	0.0047
3	−0.7665	0.0510	0.7545	0.0279
5	−0.7672	0.0365	0.7552	0.0447
6	0.4162	0.0571	−0.4097	0.0576
7	−0.7730	0.0429	0.7609	0.0378
8	0.7140	0.0472	−0.7028	0.0417
9	−0.5429	0.0358	0.5344	0.0656
10	−0.6932	0.0301	0.6823	0.0569
11	−0.5464	0.0469	0.5378	0.0582
12	0.3547	0.0080	−0.3491	0.0824
13	−0.6899	0.0401	0.6791	0.0509
14	0.5996	0.0431	−0.5902	0.0570

13.6 Boston Housing

- Moreover, cluster one corresponds to housing districts with better living quality and higher house prices, whereas cluster two corresponds to less favored districts in Boston.
- This interpretation is underlined by visual inspection of all the variables via scatterplot matrices, see, e.g., Figs. 13.17 and 13.18.

13.6 Boston Housing

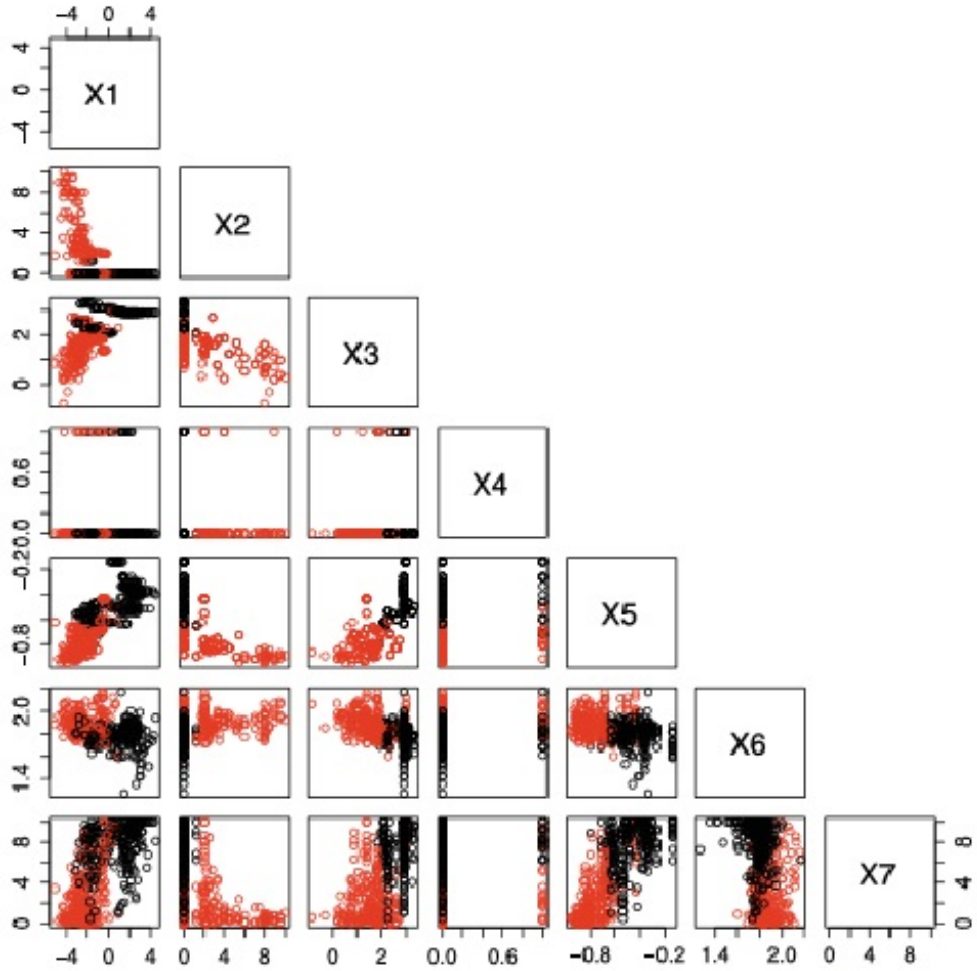


Fig. 13.17 Scatterplot matrix for variables $\widetilde{X}_1$ to $\widetilde{X}_7$ of the Boston housing data MVAclusbh

13.6 Boston Housing

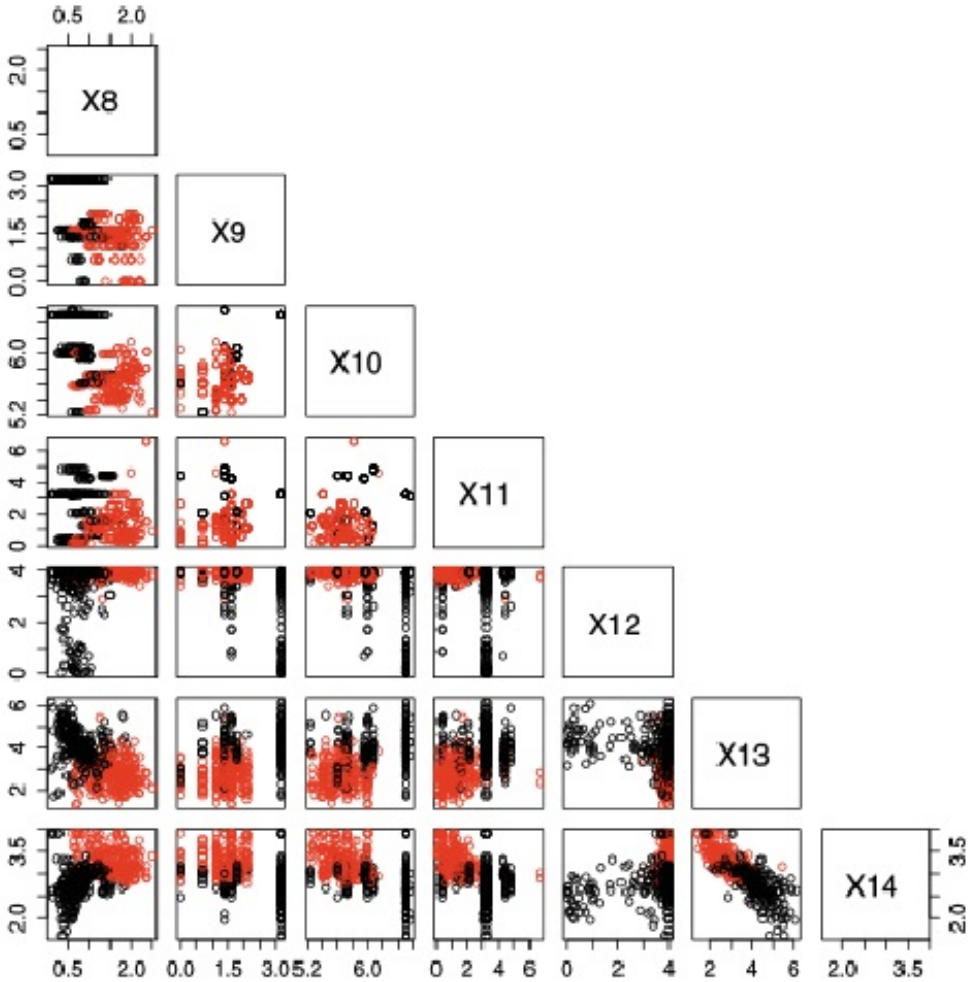


Fig. 13.18 Scatterplot matrix for variables $\tilde{X}_8$ to $\tilde{X}_{14}$ of the Boston housing data MVAclusbh

13.6 Boston Housing

- For example, the lower right boxplot of Fig. 13.15 and the correspondingly colored clusters in the last row of Fig. 13.18 confirm the role of each variable in determining the clusters. This interpretation perfectly coincides with the previous PC analysis (Fig. 11.1).
- The quality of life factor is clearly visible in Fig. 13.19, where cluster membership is distinguished by the shape and color of the points graphed according to the first two principal components.

13.6 Boston Housing

Fig. 13.19 Scatterplot of the first two PCs displaying the two clusters MVAclusbh